Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees

Alessandro Moschitti

Department of Computer Science, Systems and Production
University of Rome "Tor Vergata",
ITALY

European Conference of Machine Learning
August 17– September 22, 2006
Berlin, Germany

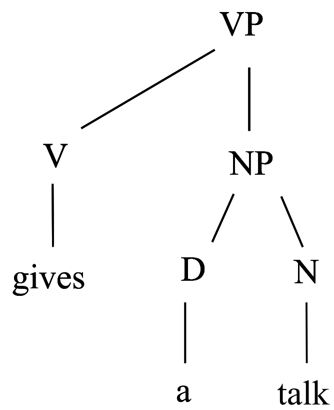
Motivations

- Modeling syntax in Natural Language learning task is complex, e.g.
  - Semantic role relations within predicate argument structures and
  - Question Classification
- Tree kernels are natural way to exploit syntactic information from sentence parse trees
  - useful to engineer novel and complex features.
- How do different tree kernels impact on different parsing paradigms and different tasks?
- Are they efficient in practical applications?

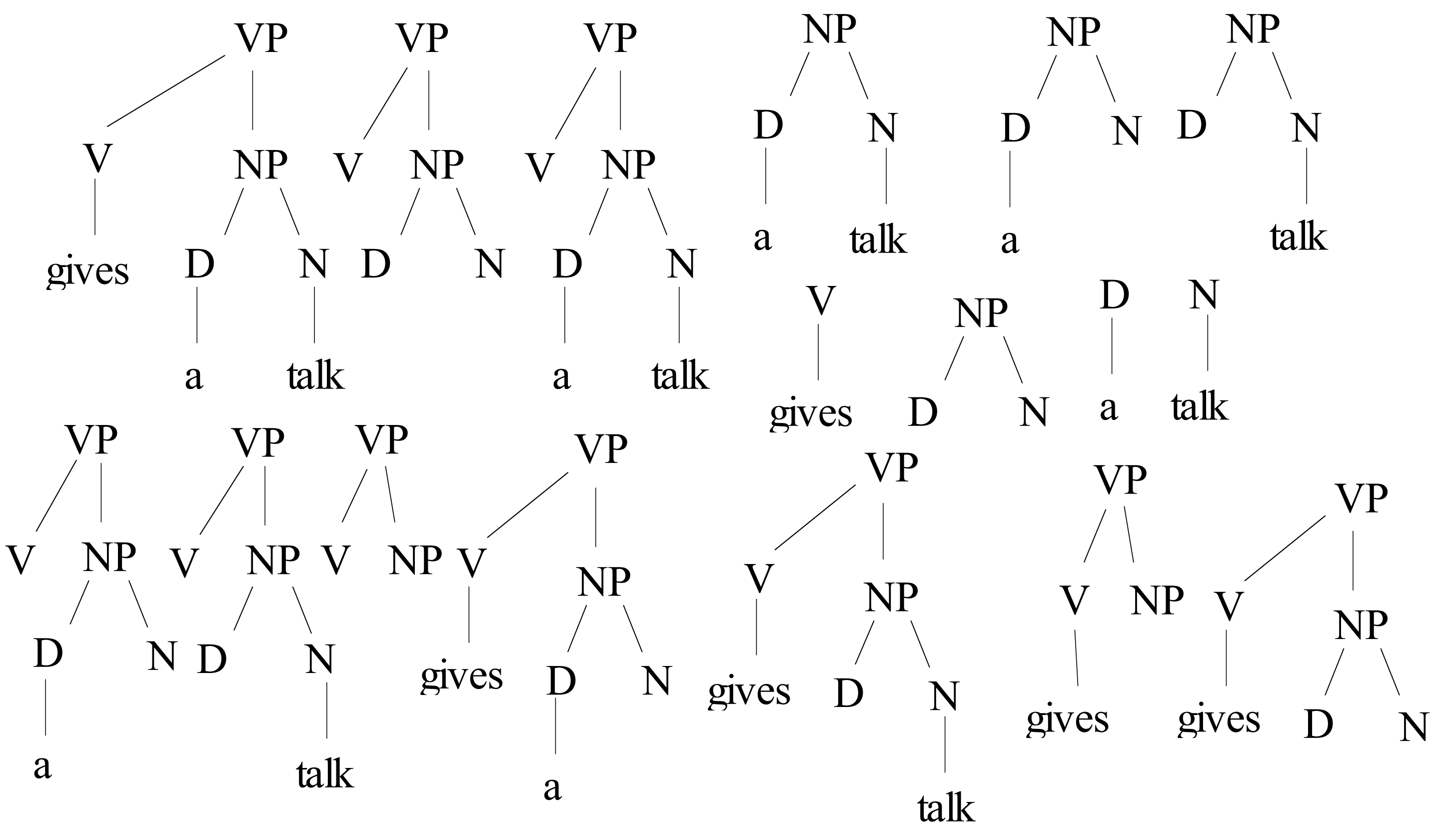
Outline

- Tree kernel types
  - Subset (SST) Tree kernel
  - Subtree (ST) kernel
  - The Partial Tree kernel
- Fast kernel algorithms
  - Efficient evaluation of PT kernel
- Two NLP applications:
  - Semantic Role Labeling
  - Question Classification
- Tree kernel Evaluations
- Conclusions

The Collins and Duffy's Tree Kernel (called SST in [Vishwanathan and Smola, 2002])

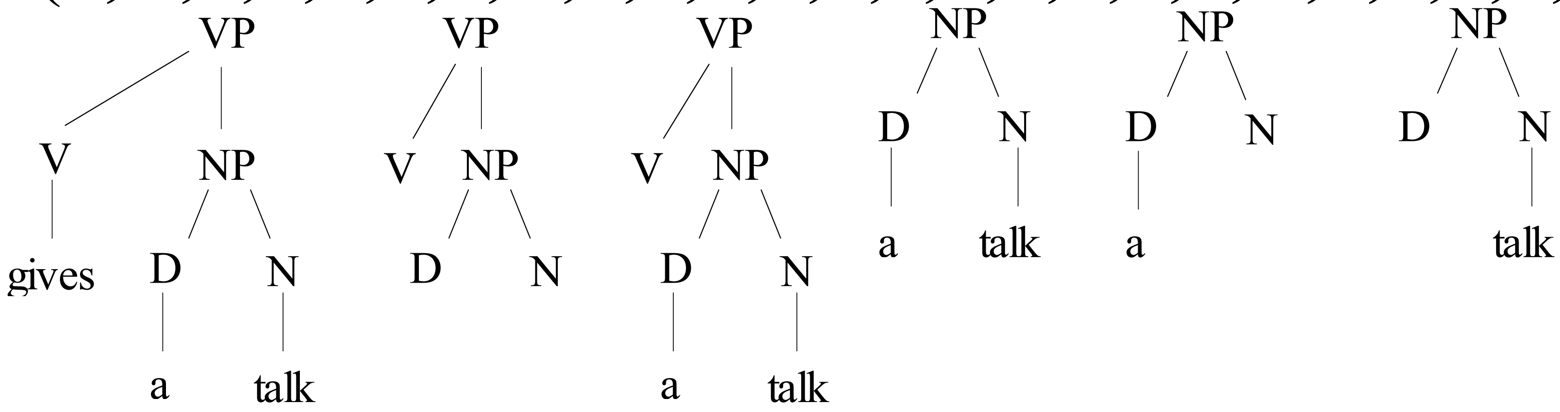


The overall fragment set



Explicit feature space

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



- $\vec{x}_1 \cdot \vec{x}_2$ counts the number of common substructures

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 Δ in O(n^2):

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

Weighting

- Decay factor

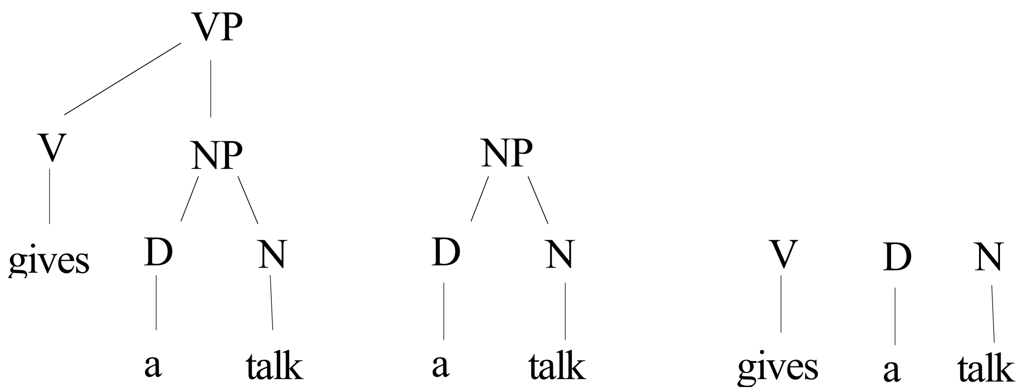
$$\Delta(n_1, n_2) = \lambda, \quad \textbf{if pre-terminals else}$$

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

- Normalization

$$K'(T_1, T_2) = \frac{K(T_1, T_2)}{\sqrt{K(T_1, T_1) \times K(T_2, T_2)}}$$

SubTree (ST) Kernel [Vishwanathan and Smola, 2002]



Evaluation

- Given the equation for the SST kernel

$$\Delta(n_1, n_2) = 0, \ \textbf{if the productions are different else}$$

$$\Delta(n_1, n_2) = 1, \ \textbf{if pre-terminals else}$$

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

Evaluation

- Given the equation for the ST kernel

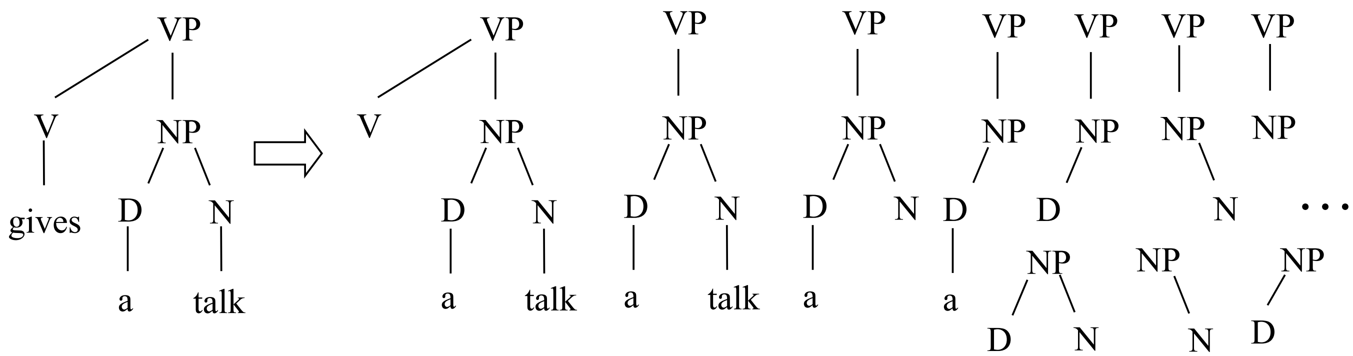
$$\Delta(n_1, n_2) = 0, \ \textbf{if the productions are different else}$$

$$\Delta(n_1, n_2) = 1, \ \textbf{if pre-terminals else}$$

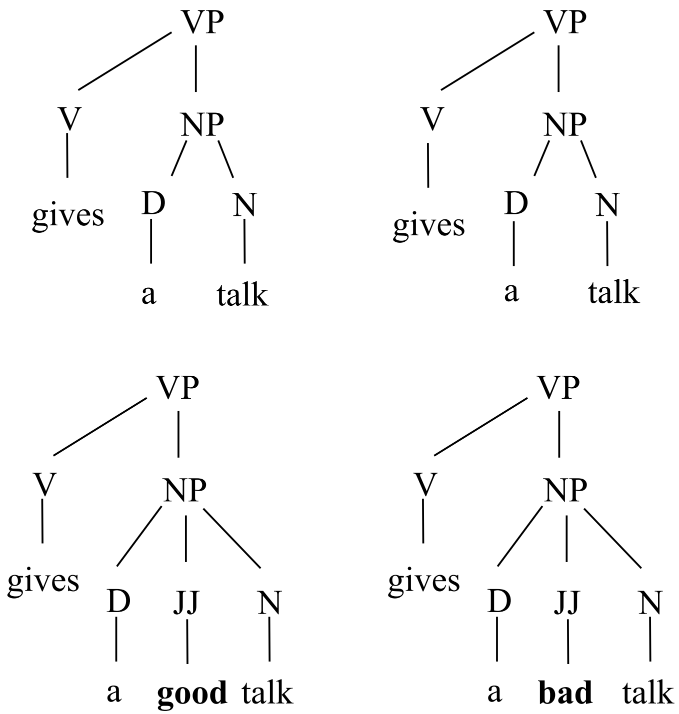
$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (\Delta(ch(n_1, j), ch(n_2, j)))$$

Labeled Ordered Tree Kernel

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



Weighting Problems



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

Partial Tree Kernel

- if the node labels of n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;

- else

$$\Delta(n_1, n_2) = 1 + \sum_{\vec{J}_1, \vec{J}_2, l(\vec{J}_1) = l(\vec{J}_2)} \prod_{i=1}^{l(\vec{J}_1)} \Delta(c_{n_1}[\vec{J}_{1i}], c_{n_2}[\vec{J}_{2i}])$$

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

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

Efficient Evaluation (2)

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

Note that D_p satisfies the recursive relation:

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

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

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:

  [$_{Arg0}$ Paul] [$_{predicate}$ gives [$_{Arg1}$ a lecture] [$_{ArgM}$ in Rome]

Semantic Role Labeling

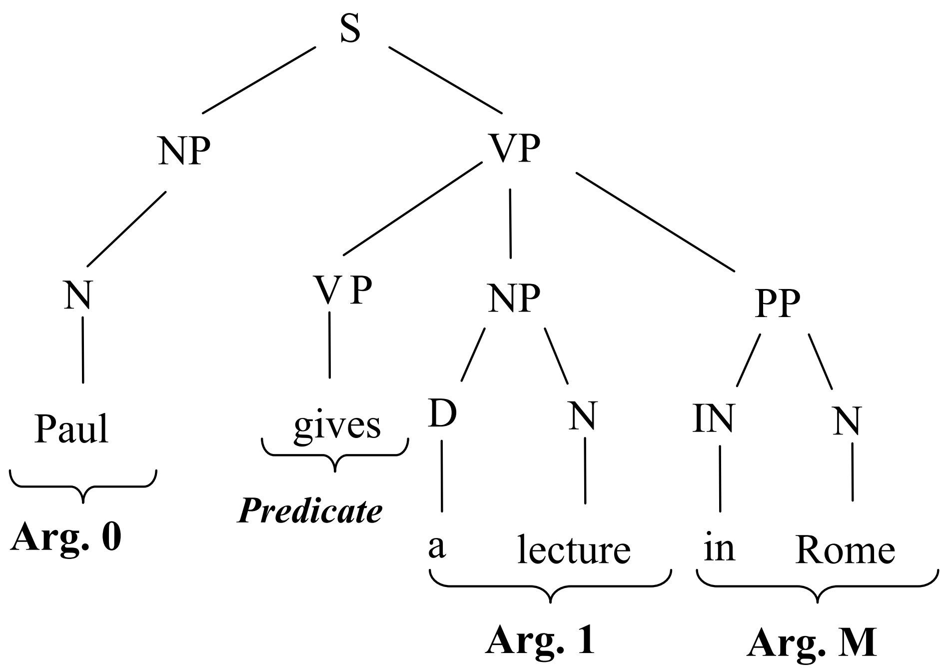
- Given an event:
  - Some words describe the relation among different participants
  - Such words can be considered predicates
  - The participants are their arguments.
- Example:

  [$_{Arg0}$ Paul] [$_{predicate}$ gives [$_{Arg1}$ a lecture] [$_{ArgM}$ in Rome]
- PropBank and FrameNet propose two different theories and resources

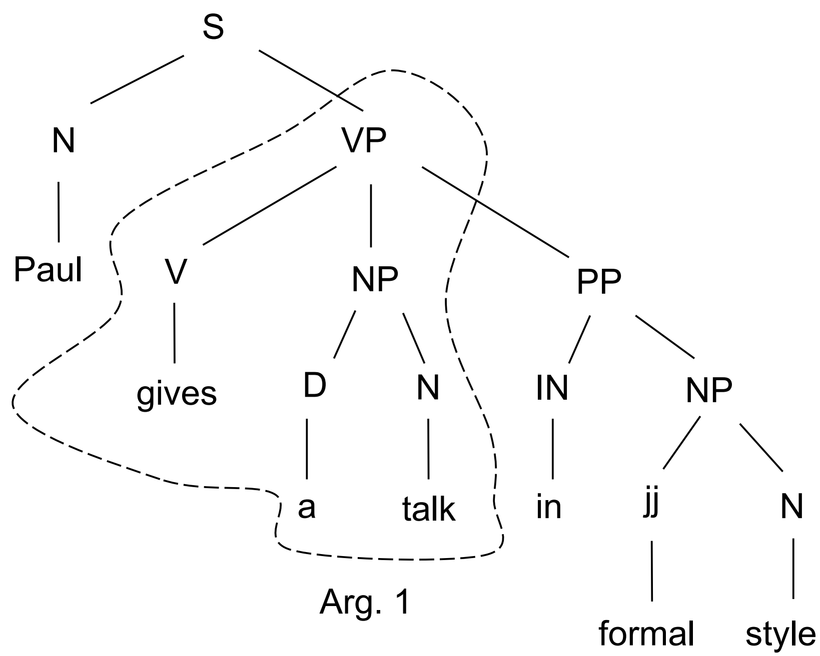
Semantic/Syntactic structures

- Given a sentence with its semantic annotation:

[$_{Arg0}$ Paul] [$_{predicate}$ gives [$_{Arg1}$ a lecture] [$_{ArgM}$ in Rome]



A Tree Kernel for Semantic Role labeling



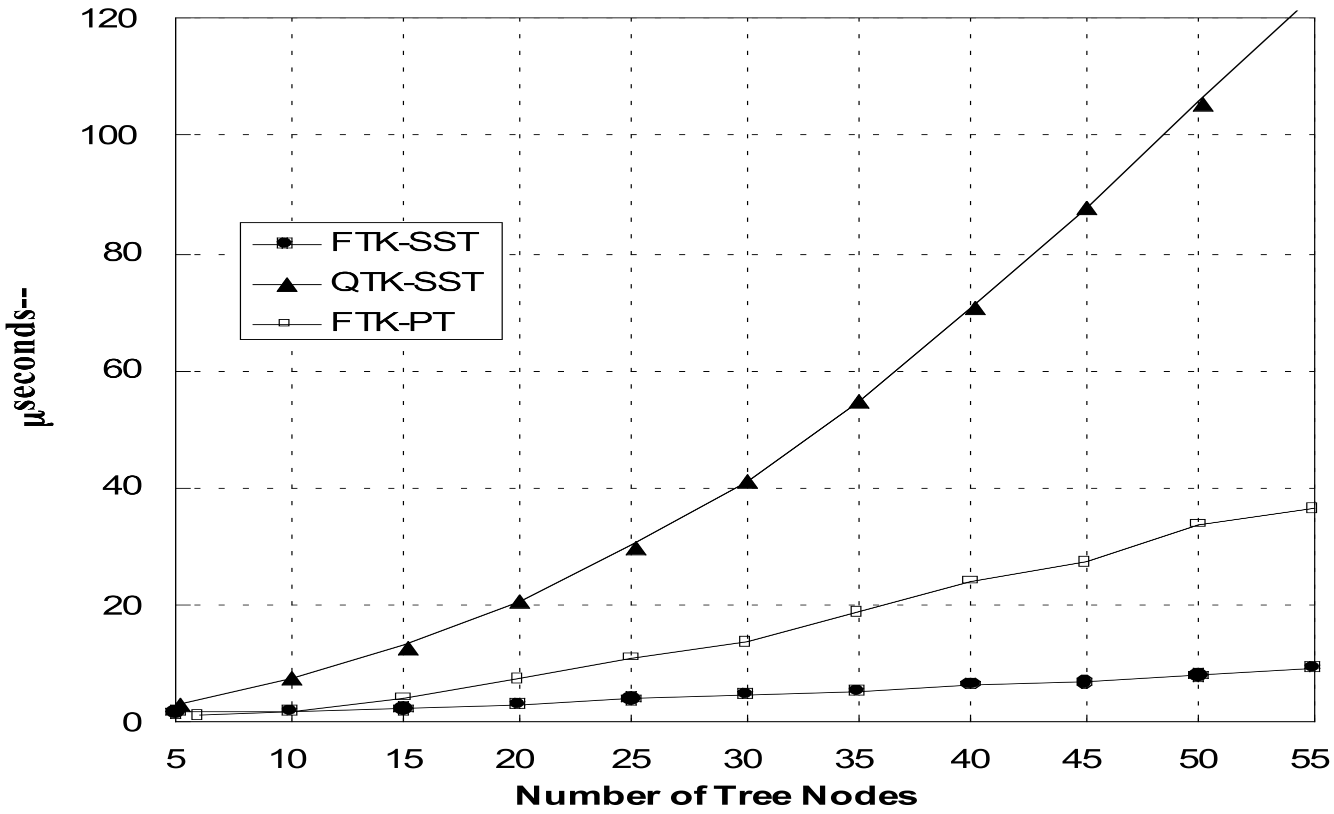
Gold Standard Tree Experiments

- 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)

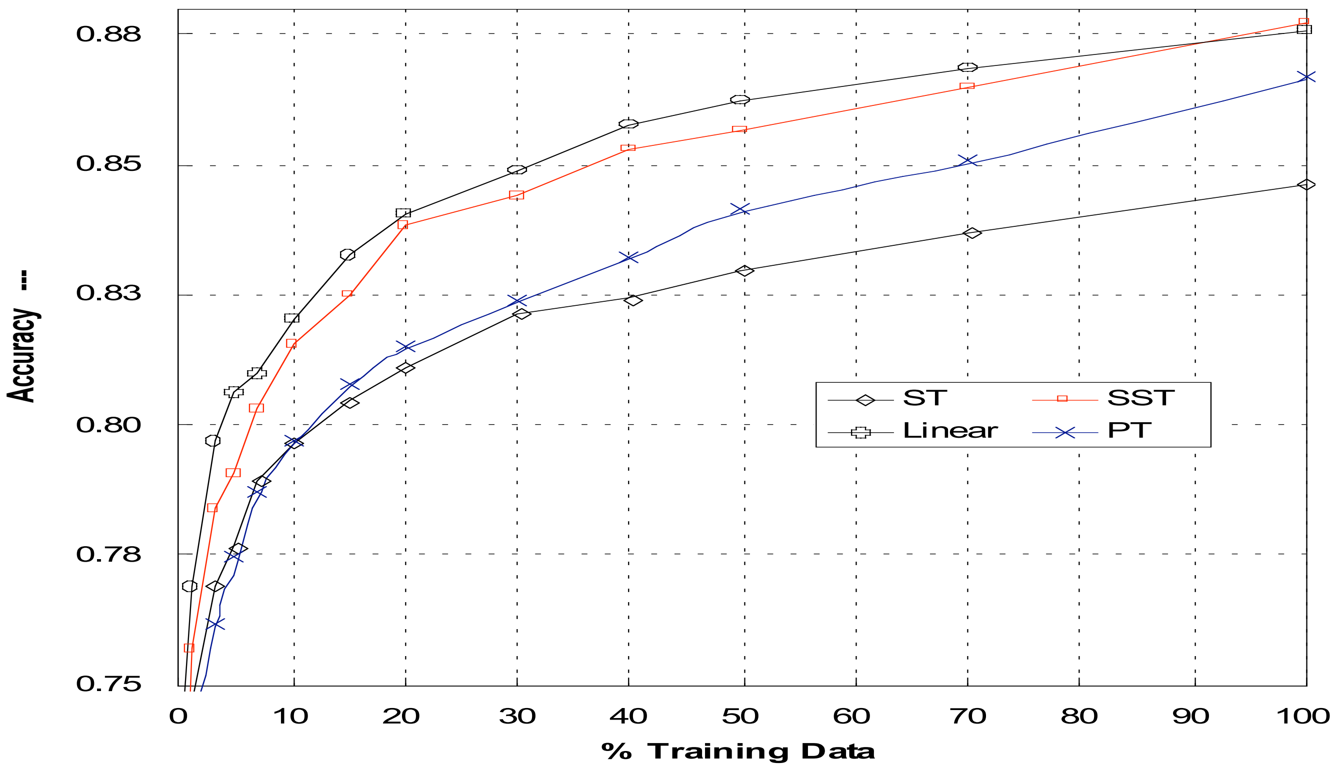
SVM-light-TK Software

- Encodes ST, SST and PT
  in SVM-light [Joachims, 1999]
- Available at http://ai-nlp.info.uniroma2.it/moschitti/
- New extensions: tree forests, vector sets and the PT kernel coming soon

Running Time of Tree Kernel Functions



Argument Classification Accuracy



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?

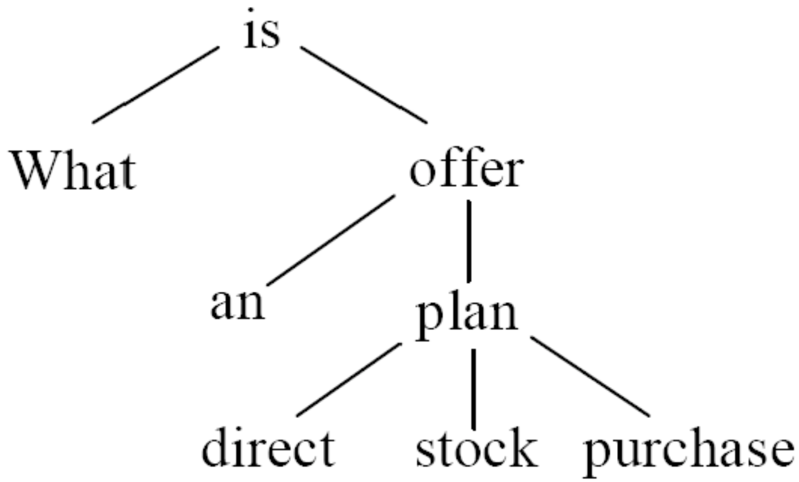
Question Classifier based on Tree Kernels

- 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?**”

The dependency Tree

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



- PTs can be very effective, e.g.

  [Plan [direct][purchase]] and [Plan [stock][purchase]]

Question Classification results

Parsers	Constituent		Dependency		BOW
Kernels	SST	PT	SST	PT	Linear
Acc.	88.2	87.2	82.1	90.4	86.3

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

Riferimenti

- Scaricabili dalla rete:
  - A. Moschitti, *A study on Convolution Kernels for Shallow Semantic Parsing*. In proceedings of the 42-th Conference on Association for Computational Linguistic (ACL-2004), Barcelona, Spain, 2004.
  - A. Moschitti, *Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees*. In Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany, 2006.
  - M. Collins and N. Duffy, 2002, *New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron*. In ACL02, 2002.
  - S.V.N. Vishwanathan and A.J. Smola. *Fast kernels on strings and trees*. In Proceedings of Neural Information Processing Systems, 2002.

PropBank

- Levin proposed a classification of verbs according to their syntactic use (syntactic alternations)
- The same verb can be member of different classes
- In PropBank argument labels are assigned to verb's entries depending on the Levin's class.
- ARG 0,1..,n are such semantic roles.

FrameNet

- Based on Fillmore's theory regarding frame semantics
- FrameNet corpus is divided in frames each having examples annotated with frame specific roles

  Ex:

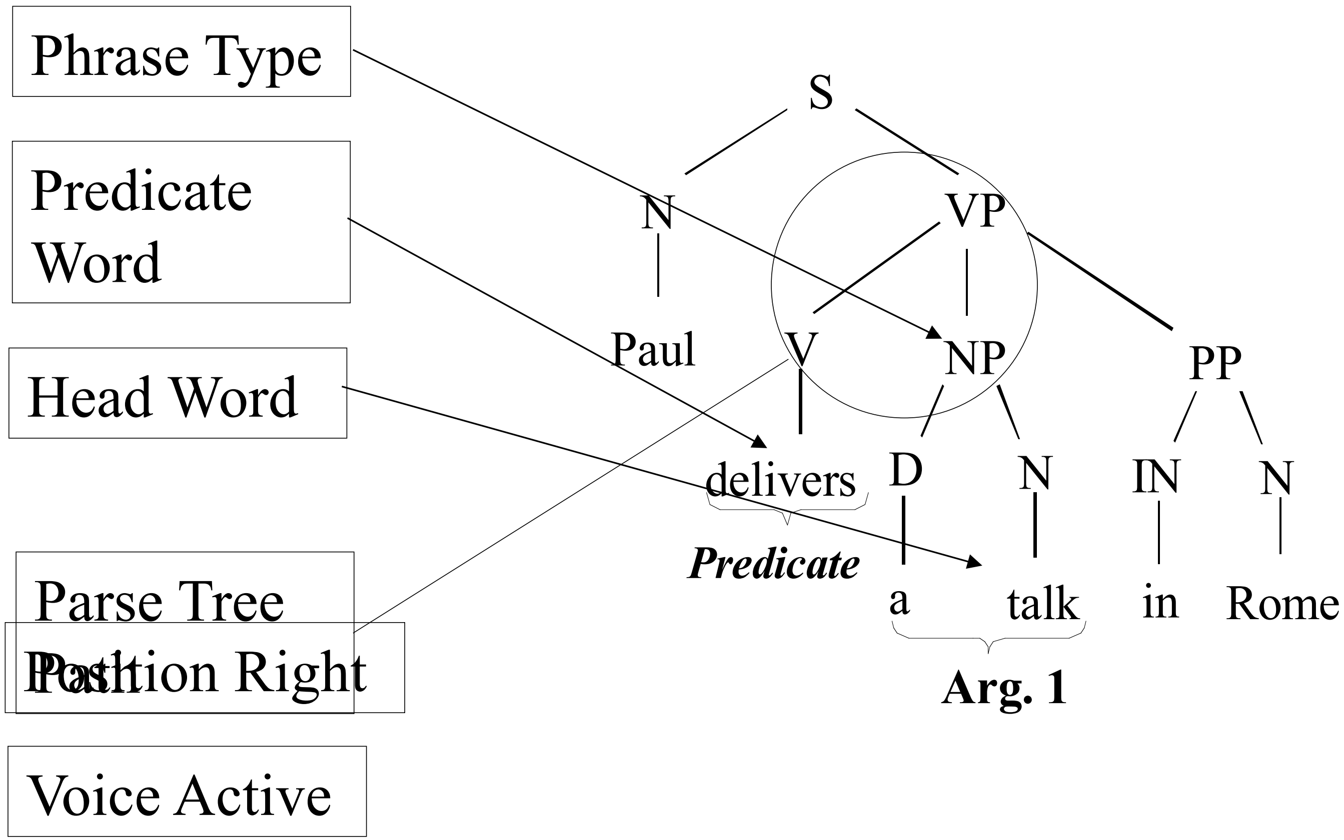
  [$_{Speaker}$ John] [$_{Predicate}$ told] [$_{Addressee}$ Mary] [$_{Message}$ to shut the door].

Semantic/Syntactic structures

- Given a sentence with its semantic annotation:

[$_{Arg0}$ Paul] [$_{predicate}$ gives [$_{Arg1}$ a lecture] [$_{ArgM}$ in Rome]

An example



Phrase Type
Predicate Word
Head Word
Parse Tree Path
Position Right
Voice Active
S
N
VP
Paul
V
NP
PP
delivers
D
N
IN
N
Predicate
a
talk
in
Rome
Arg. 1

Automatic Predicate Argument Extraction

- Boundary Detection
  - One binary classifier
- Argument Type Classification
  - Multi-classification problem
  - *n* binary classifiers (ONE-vs-ALL)
  - Select the argument with maximum score

Typical standard flat features
(Gildea & Jurasfky, 2002)

- Phrase Type of the argument
- Parse Tree Path, between the predicate and the argument
- Head word
- Predicate Word
- Position
- Voice

Flat features (Linear Kernel)

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

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

PT PTP HW PW P V

- The dot product counts the number of features in common

$$\vec{x} \cdot \vec{z}$$

Automatic Predicate Argument Extraction

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

PropBank

- 300.000-word corpus of Wall Street Journal articles
- The annotation is ***based on*** the Levin's classes.
- The arguments range from Arg0 to Arg9, ArgM.
- Lower numbered arguments more regular e.g.
  - Arg0 → subject and Arg1 → direct object.
- Higher numbered arguments are less consistent
  - assigned per-verb basis.

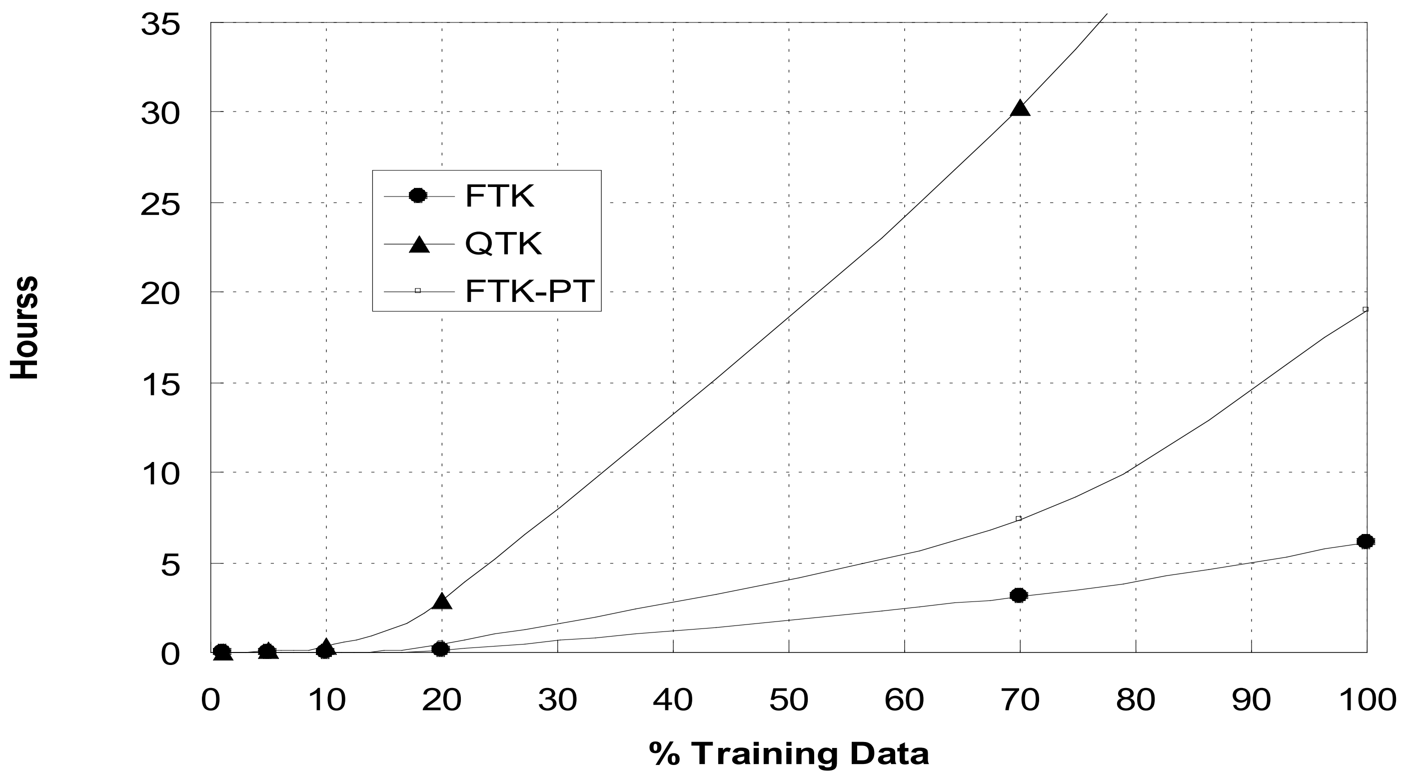
Evaluation

- Given the equation for the SST kernel

$$\Delta(n_1, n_2) = 0, \textbf{ if the productions are different else}$$
$$\Delta(n_1, n_2) = 1, \textbf{ if pre-terminals else}$$
$$\Delta(n_1, n_2) = S_{n_1, n_2}(nc(n_1), nc(n_2))$$

$$\begin{aligned} S_{n_1,n_2}(i,j) &= S_{n_1,n_2}(i-1,j) + S_{n_1,n_2}(i,j-1) \\ &\quad - S_{n_1,n_2}(i-1,j-1) \\ &\quad + S_{n_1,n_2}(i-1,j-1) \cdot \Delta(ch(n1,i), ch(n2,j)) \end{aligned}$$

SVM Learning Time



Faster evaluation of kernels

$$K(T_1,T_2) = \sum_{\langle n_1,n_2\rangle \in NP} \Delta(n_1,n_2)$$

$$NP = \{\langle n_1,n_2\rangle \in T_1 \times T_2 : \Delta(n_1,n_2) \neq 0\} =$$
$$= \{\langle n_1,n_2\rangle \in T_1 \times T_2 : P(n_1) = P(n_2)\},$$

where $P(n_1)$ and $P(n_2)$ are the production rules used at nodes n_1 and n_2

Observations

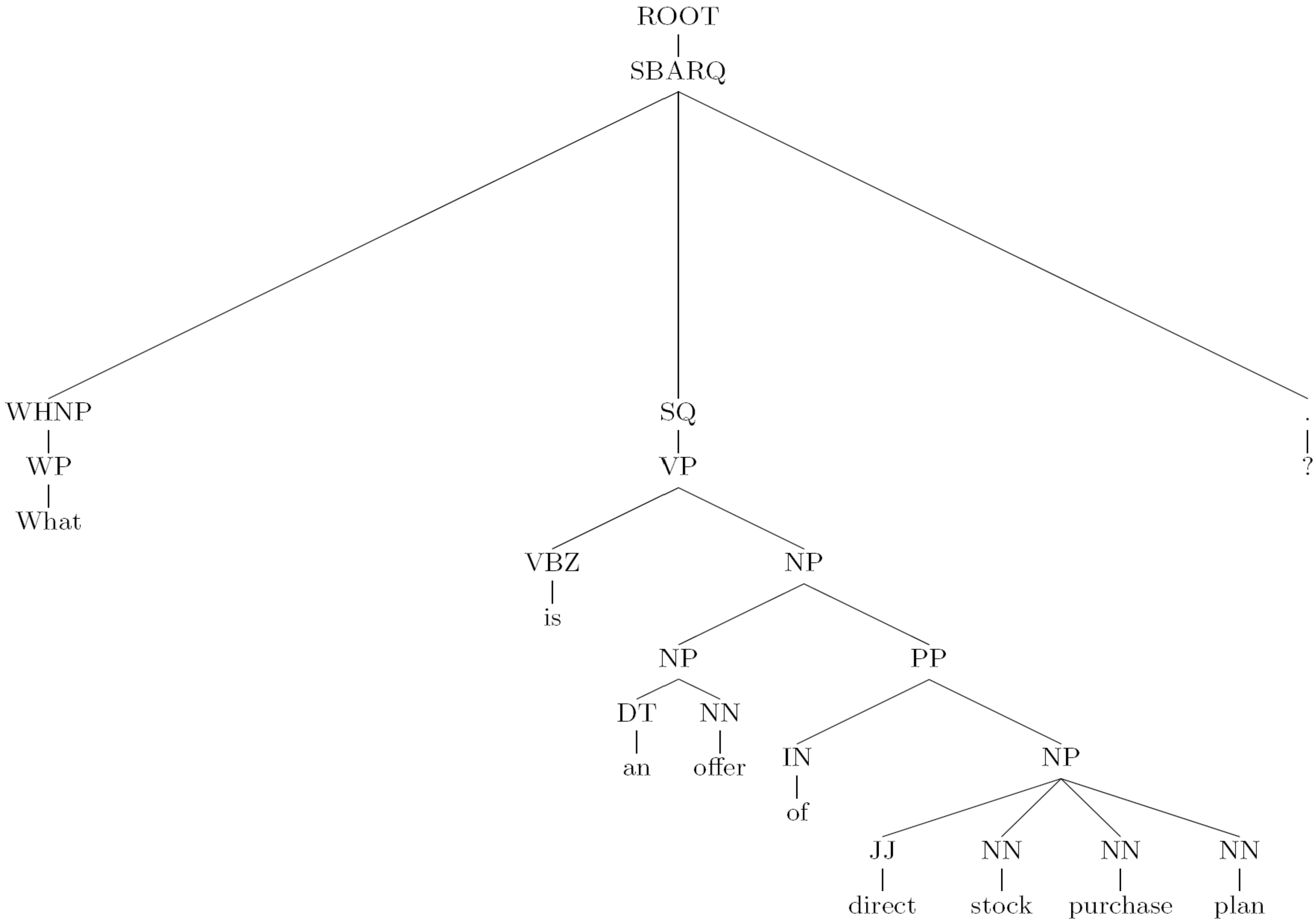
- We can sort the production rules used in T_1 and T_2, at loading time
- At learning time we may evaluate NP in

  $|T_1|+|T_2|$ *running time*
- If T_1 and T_2 are generated by only one production rule $\Rightarrow O(|T_1| \times |T_2|)\ldots$

Automatic SRL approach

- Boundary detection: detect parse tree nodes that corresponds to an argument
- Argument Classification: classify the type of each argument node
- For both task the same approach and almost the same features are used…

  ….we focus on classification only


ROOT
SBARQ
WHNP
SQ
.
WP
VP
?
What
VBZ
NP
is
NP
PP
DT
NN
IN
NP
an
offer
of
JJ
NN
NN
NN
direct
stock
purchase
plan