Paper Discussion: Convolution Kernels for Natural Language (Collins & Duffy '02)

Huizhen Yu

janey.yu@cs.helsinki.fi
Dept. Computer Science, Univ. of Helsinki

PSD Seminar, Feb. 07, 2008

Natural Language Task Examples

Map strings to *hidden structures*:

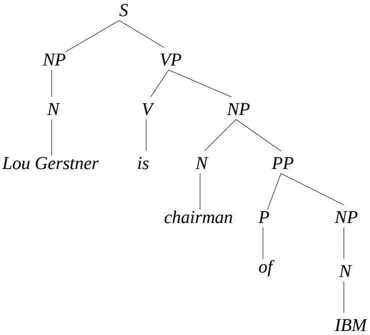
- named identity boundaries

Lou	*Gerstner*	*is*	*chairman*	*of*	*IBM*
SP	*CP*	*N*	*N*	*N*	*SC*

- part-of-speech tags

Lou	*Gerstner*	*is*	*chairman*	*of*	*IBM*
N	*N*	*V*	*N*	*P*	*N*

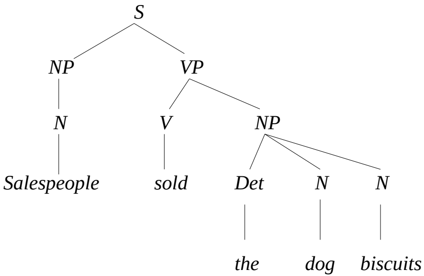
- parse tree

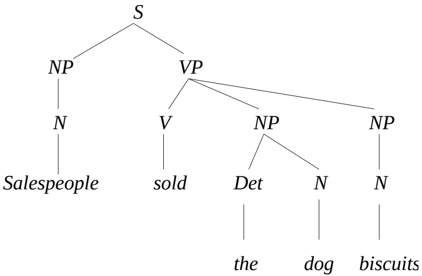


Ambiguities in Parsing

E. Charniak, *Statistical Language Learning*, p. 7:

"Salespeople sold the dog biscuits." $\rightarrow$





Kernels on Parse Trees

- A finite set of all admissible subtrees $\{s_1, \ldots, s_n\}$
- Kernel on trees from feature mapping $h : \{T\} \to \Re^n$,

$$h(T) = \big(h_1(T), \ldots, h_n(T)\big), \qquad h_i(T) = \#\text{occurences of } s_i \text{ in } T$$

$$\begin{aligned} K(T_1, T_2) &= h(T_1) \cdot h(T_2) \\ &= \sum_{i=1}^{n} \left(\sum_{n_1 \in N_1} I_i(T_1(n_1)) \right) \left(\sum_{n_2 \in N_2} I_i(T_2(n_2)) \right) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \underbrace{\left(\sum_{i=1}^{n} I_i(T_1(n_1))\, I_i(T_2(n_2)) \right)}_{\stackrel{def}{=} C(n_1, n_2)} \end{aligned}$$

$C(\cdot,\cdot)$: kn. on subtrees
$\leftarrow K(\cdot,\cdot)$ in conv.-kn. form

where for $j = 1, 2$,
N_j: the set of (non-terminal) nodes of T_j,
$T_j(m)$: the subtree of T_j rooted at node m,
$I_i(T(m))$: the indicator function for s_i seen rooted at node m of T.

Kernels on Parse Trees

- A finite set of all admissible subtrees $\{s_1, \ldots, s_n\}$
- Kernel on trees from feature mapping $h : \{T\} \to \Re^n$,

$$h(T) = \big(h_1(T), \ldots, h_n(T)\big), \qquad h_i(T) = \#\text{occurences of } s_i \text{ in } T$$

$$\begin{aligned} K(T_1, T_2) &= h(T_1) \cdot h(T_2) \\ &= \sum_{i=1}^{n} \left(\sum_{n_1 \in N_1} I_i(T_1(n_1)) \right) \left(\sum_{n_2 \in N_2} I_i(T_2(n_2)) \right) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \underbrace{\left(\sum_{i=1}^{n} I_i(T_1(n_1))\, I_i(T_2(n_2)) \right)}_{\stackrel{def}{=} C(n_1, n_2)} \end{aligned}$$

$C(\cdot,\cdot)$: kn. on subtrees
$\leftarrow K(\cdot,\cdot)$ in conv.-kn. form

where for $j = 1, 2$,
N_j: the set of (non-terminal) nodes of T_j,
$T_j(m)$: the subtree of T_j rooted at node m,
$I_i(T(m))$: the indicator function for s_i seen rooted at node m of T.

Kernels on Parse Trees

- A finite set of all admissible subtrees $\{s_1, \ldots, s_n\}$
- Kernel on trees from feature mapping $h : \{T\} \to \Re^n$,

$$h(T) = \big(h_1(T), \ldots, h_n(T)\big), \qquad h_i(T) = \#\text{occurences of } s_i \text{ in } T$$

$$\begin{aligned} K(T_1, T_2) &= h(T_1) \cdot h(T_2) \\ &= \sum_{i=1}^{n} \left(\sum_{n_1 \in N_1} I_i(T_1(n_1)) \right) \left(\sum_{n_2 \in N_2} I_i(T_2(n_2)) \right) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \underbrace{\left(\sum_{i=1}^{n} I_i(T_1(n_1))\, I_i(T_2(n_2)) \right)}_{\stackrel{def}{=} C(n_1, n_2)} \end{aligned}$$

$C(\cdot,\cdot)$: kn. on subtrees
$\leftarrow K(\cdot,\cdot)$ in conv.-kn. form

where for $j = 1, 2$,
N_j: the set of (non-terminal) nodes of T_j,
$T_j(m)$: the subtree of T_j rooted at node m,
$I_i(T(m))$: the indicator function for s_i seen rooted at node m of T.

Kernels on Parse Trees

- A finite set of all admissible subtrees $\{s_1, \ldots, s_n\}$
- Kernel on trees from feature mapping $h : \{T\} \to \Re^n$,

$$h(T) = \big(h_1(T), \ldots, h_n(T)\big), \qquad h_i(T) = \#\text{occurences of } s_i \text{ in } T$$

$$\begin{aligned} K(T_1, T_2) &= h(T_1) \cdot h(T_2) \\ &= \sum_{i=1}^{n} \left(\sum_{n_1 \in N_1} I_i\big(T_1(n_1)\big) \right) \left(\sum_{n_2 \in N_2} I_i\big(T_2(n_2)\big) \right) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \underbrace{\left(\sum_{i=1}^{n} I_i\big(T_1(n_1)\big)\, I_i\big(T_2(n_2)\big) \right)}_{\stackrel{def}{=} C(n_1, n_2)} \end{aligned}$$

$C(\cdot,\cdot)$: kn. on subtrees
$\leftarrow K(\cdot,\cdot)$ in conv.-kn. form

where for $j = 1, 2$,
N_j: the set of (non-terminal) nodes of T_j,
$T_j(m)$: the subtree of T_j rooted at node m,
$I_i(T(m))$: the indicator function for s_i seen rooted at node m of T.

Kernels on Parse Trees

- A finite set of all admissible subtrees $\{s_1, \ldots, s_n\}$
- Kernel on trees from feature mapping $h : \{T\} \to \Re^n$,

$$h(T) = (h_1(T), \ldots, h_n(T)), \qquad h_i(T) = \#\text{occurences of } s_i \text{ in } T$$

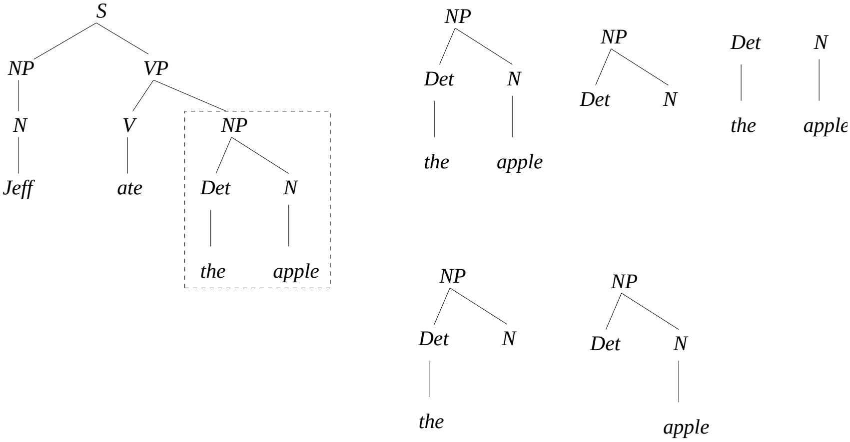
$$\begin{aligned}
K(T_1, T_2) &= h(T_1) \cdot h(T_2) \\
&= \sum_{i=1}^{n} \left(\sum_{n_1 \in N_1} I_i(T_1(n_1)) \right) \left(\sum_{n_2 \in N_2} I_i(T_2(n_2)) \right) \\
&= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \underbrace{\left(\sum_{i=1}^{n} I_i(T_1(n_1))\, I_i(T_2(n_2)) \right)}_{\stackrel{def}{=} C(n_1, n_2)}
\end{aligned}$$

$C(\cdot,\cdot)$: kn. on subtrees
$\leftarrow K(\cdot,\cdot)$ in conv.-kn. form

where for $j = 1, 2$,
N_j: the set of (non-terminal) nodes of T_j,
$T_j(m)$: the subtree of T_j rooted at node m,
$I_i(T(m))$: the indicator function for s_i seen rooted at node m of T.

Kernel Evaluation: Recursive Computation (I)

- Examples of *admissible* subtrees (tree fragments, a.-subtrees)



- Include the complete rule productions (prod) at the node
- The full set of a.-subtrees need not be stored explicitly

Kernel Evaluation: Recursive Computation (II)

- $K(T_1, T_2) = \sum_{n_1 \in N_1, n_2 \in N_2} C(n_1, n_2)$

  $C(n_1, n_2)$: #common a.-subtrees rooted at n_1, n_2

- Compute $C(n_1, n_2)$ recursively:

$$C(n_1, n_2) = \begin{cases} 0, & \text{prod}(n_1) \neq \text{prod}(n_2) \\ 1, & \text{prod}(n_1) = \text{prod}(n_2), \text{\& } n_1, n_2 \text{ pre-terminals} \end{cases}$$

$$C(n_1, n_2) = \prod_{j=1}^{nc(n_1)} \left(1 + C\Big(\underbrace{ch(n_1, j)}_{jth\ child}, ch(n_2, j) \Big) \right),$$

$$\text{if } \text{prod}(n_1) = \text{prod}(n_2), \text{\& } n_1, n_2 \text{ not pre-terminals}$$

  (recursive comp. is in part due to the definition of a.-subtrees)

- Complexity of computing $k(T_1, T_2)$: $O(|N_1||N_2|)$, in practice, linear

Practical Concerns and Modifications of Kernels

- Tree size has too much effect on kernel; do normalization:

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

- Extreme values between similar and dissimilar trees:

$$K(T_1, T_2) \approx 10^6, \quad T_1 \sim T_2, \qquad K(T_1, T_2) \approx 10, \quad T_1 \not\sim T_2$$

  kernel is too “peaked;” prediction will be close to “nearest neighbor” rule
  Solution? (“radialization” – $\ln K$ – didn’t help.)
- Modifications: downweight large a.-subtrees
  - option 1: restrict the depth of a.-subtrees; recursive evaluation is still possible by computing $C(n_1, n_2, d)$
  - option 2: scale the relative importance of a.-subtrees by their size

$$K'(T_1, T_2) = \sum_{i=1}^{n} \lambda^{size(s_i)} h_i(T_1)\, h_i(T_2)$$

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

Parsing Modeled as Re-ranking

- Training examples: (s_i, t_i), sentence and correct parse tree pairs
- Create data: input s + candidate parses, e.g., $\mathcal{C}(s) = \{T_1, \ldots, T_{100}\}$
- Parse/prediction output for s:

$$\arg\max_{T \in \mathcal{C}(s)} w \cdot h(T) = \arg\max_{T \in \mathcal{C}(s)} \sum_j \alpha_j K(T, \widehat{T}_j)$$

- Criterion for optimizing α: rank t_i as the highest-score parse tree for s_i
- Training algorithm: a variant of voted perceptron

Parsing Experiments on Penn Treebank ATIS Corpus

- Treebank is randomly split (10 ways) into training (800), development (200) and test (336) sets.
- PCFG, trained on training set, produces 100 candidates for each sentence; use 20 candidates per sentence in training.
- Test: choose the best tree from the 100 candidates.
- Parse score: measure precision and recall
  constituent: a non-terminal label and its span
  c_i : #correctly placed constituents in the ith test tree
  p_i : #constituents proposed
  g_i : #constituents in the true parse tree

$$score = 100\% \times \frac{1}{\sum_i g_i} \sum_i g_i \times \frac{1}{2}\left(\frac{c_i}{p_i} + \frac{c_i}{g_i}\right)$$

$$rel. - improvement = 100\% \times \frac{score_{perc} - score_{PCFG}}{100 - score_{PCFG}}$$

- PCFG scored 74

Parsing Experiments on Penn Treebank ATIS Corpus (Cont'd)

- Varying the maximum depth of a.-subtrees

depth	1	2	3	4	5	6
score	73 ± 1	79 ± 1	80 ± 1	79 ± 1	79 ± 1	78 ± 0.01
improvement	-1 ± 4	20 ± 6	23 ± 3	21 ± 4	19 ± 4	18 ± 3

- Varying the scale parameter λ for the size of a.-subtrees

λ	0.1	0.2	0.3	0.4	0.5	0.6
score	77 ± 1	78 ± 1	79 ± 1	79 ± 1	79 ± 1	79 ± 1
improvement	11 ± 6	17 ± 5	20 ± 4	21 ± 3	21 ± 4	22 ± 4

λ	0.7	0.8	0.9
score	79 ± 1	79 ± 1	78 ± 1
improvement	21 ± 4	19 ± 4	17 ± 5

Discussions

- Compact representation of $f(T) = \sum_j \alpha_j K(T, \widehat{T}_j)$:
  find common a.-subtrees in $\widehat{T}_j$, add weights together, and make a weighted acyclic graph
- A re-ranking model for structured prediction:

$$\arg\max_{T \in \mathcal{T}} f(T)$$

  Other options?
- Kernel on joint input-output space

Discussions

- Compact representation of $f(T) = \sum_j \alpha_j K(T, \widehat{T}_j)$:
  find common a.-subtrees in $\widehat{T}_j$, add weights together, and make a weighted acyclic graph
- A re-ranking model for structured prediction:

  $$\arg\max_{T \in \mathcal{T}} f(T)$$

  Other options?
- Kernel on joint input-output space

Discussions

- Compact representation of $f(T) = \sum_j \alpha_j K(T, \widehat{T}_j)$:
  find common a.-subtrees in $\widehat{T}_j$, add weights together, and make a weighted acyclic graph
- A re-ranking model for structured prediction:

  $$\arg\max_{T \in \mathcal{T}} f(T)$$

  Other options?
- Kernel on joint input-output space