Chapter 4 Classic Parsing algorithms

Part two: Inside-Outside Algorithm

- Inference of SCFG
- Learning of SCFG
- Examples:
  Tangram and hierarchical tiling for scene and human attributes.

1, Review: grammar of strings in NLP

A probabilistic context-free grammar (PCFG) has four components

- A set Σ of terminal symbols
  - the vocabulary of a language
- A set N of nonterminal symbols
- A start symbol $S \in N$
- A set R of production rules
  - Each specifies how a nonterminal can be rewritten to string of terminals and/or nonterminals

Grammar of Strings

A grammar specifies how to generate a sentence

- starting from a string containing only the start symbol S
- recursively applying the rules to rewrite the string
- until the string contains only terminals

The generative process specifies the grammatical structure of a sentence.

Probabilistic Grammars

Each rule is associated with a probability

$$\alpha \to \beta : P(\alpha \to \beta|\alpha)$$

A probabilistic grammar defines a joint probability of a grammatical structure y and its sentence x

$$P(x, y|G) = \prod_{r \in R} \theta_r{}^{f_r(x,y)}$$

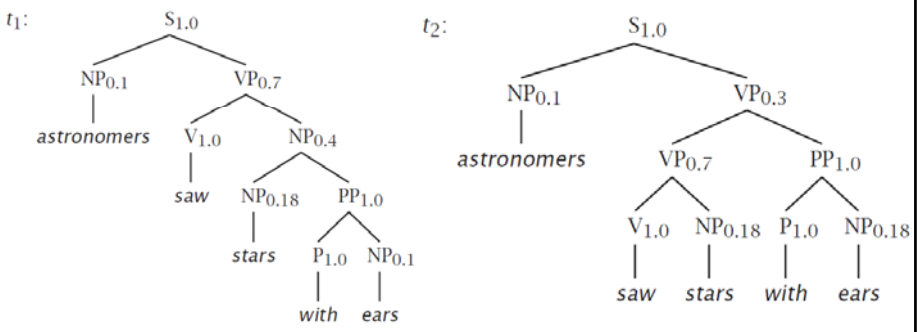
θ_r is the probability of rule r.
$f_r(x, y)$ is the number of times rule r is used in generating x and y.

An Example

S → NP VP	1.0	NP → NP PP	0.4
PP → P NP	1.0	NP → *astronomers*	0.1
VP → V NP	0.7	NP → *ears*	0.18
VP → VP PP	0.3	NP → *saw*	0.04
P → *with*	1.0	NP → *stars*	0.18
V → *saw*	1.0	NP → *telescopes*	0.1

Examples from [Christopher D. Manning and Hinrich Schütze]

An Example: a sentence has multiple parses

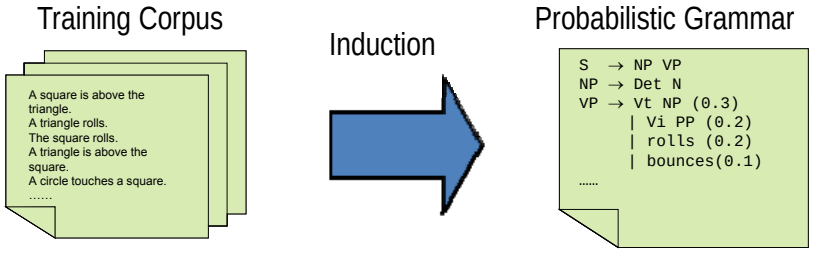


$$P(t_1) = 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 1.0 \times 0.18 = 0.0009072$$

$$P(t_2) = 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 1.0 \times 0.18 = 0.0006804$$

$$P(w_{15}) = P(t_1) + P(t_2) = 0.0015876$$

2, Learning a grammar from a corpus



Supervised Methods

– Rely on a training corpus of sentences annotated with grammatical structures (parses)

Unsupervised Methods

– Do not require annotated data

Learning a grammar from a corpus

Structure learning

– Try to find an optimal set of grammar rules

Parameter learning

– Assume a fixed set of grammar rules and try to learn their probabilities

$$\Theta^* = \arg\max_{\Theta} P(X|\Theta)$$

Structure learning can be transformed to a parameter learning problem, such as the dependency grammar and Tangram model.

3, Inside-outside algorithm for inference

Assume the grammar is in the Chomsky normal form (CNF)

– Only two types of rules

$N_1 \rightarrow N_2 N_3$ Binary split

$N_1 \rightarrow w$ Terminating

Each node N can be split in many ways.

This is what the Tangram model does in And-or graph.

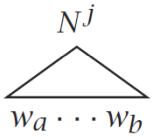
Inside-outside algorithm

Notations

Sentence: sequence of words $w_1 \cdots w_m$

w_{ab}: the subsequence $w_a \cdots w_b$

N^i_{ab}: nonterminal N^i dominates $w_a \cdots w_b$



$N^i \stackrel{*}{\Longrightarrow} \zeta$: Repeated derivation from N^i gives ζ.

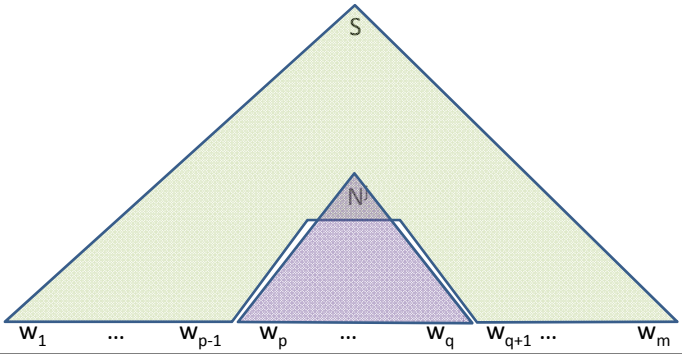
Inside-outside algorithm

It simultaneously computes the "detection probabilities" for all nodes in the grammar.

Two types of probabilities

Outside = $\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m}|G)$

Inside = $\beta_j(p,q) = P(w_{pq}|N^j_{pq}, G)$



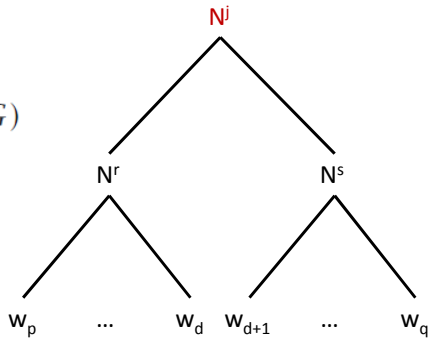
Computing inside probabilities

Base case

$$\beta_j(k,k) = P(w_k|N^j_{kk}, G)$$
$$= P(N^j \rightarrow w_k|G)$$

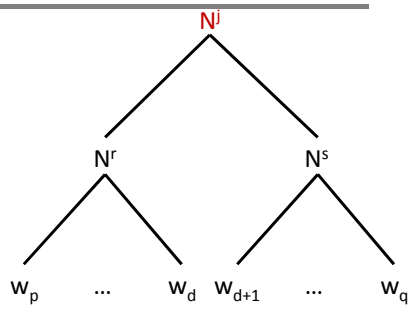
Bottom-up recursion

$$\beta_j(p,q) = P(w_{pq}|N^j_{pq}, G)$$



Computing inside probabilities

Bottom-up recursion



$$\begin{aligned}\beta_j(p,q) &= P(w_{pq}|N^j_{pq},G)\\ &= \sum_{r,s}\sum_{d=p}^{q-1} P(N^r_{pd},N^s_{(d+1)q}|N^j_{pq},G)\\ &\qquad P(w_{pd}|N^r_{pd},G)P(w_{(d+1)q}|N^s_{(d+1)q},G)\\ &= \sum_{r,s}\sum_{d=p}^{q-1} P(N^j\rightarrow N^rN^s)\beta_r(p,d)\beta_s(d+1,q)\end{aligned}$$

Computing outside probabilities

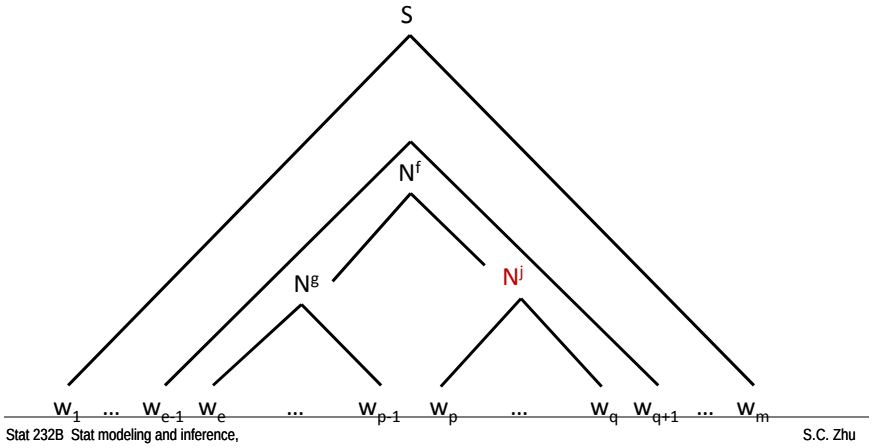
Base case

$$\alpha_s(1,m) = 1$$
$$\alpha_j(1,m) = 0, \text{ for } j \neq s$$

Computing outside probabilities

Top-down recursion

$$\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m}|G)$$



Computing outside probabilities

Top-down recursion

$$\begin{aligned}
\alpha_j(p,q) &= P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m}|G) \\
&= [\sum_{f,g}\sum_{e=q+1}^{m} P(w_{1(p-1)}, w_{(e+1)m}, N^f_{pe}) P(N^j_{pq}, N^g_{(q+1)e}|N^f_{pe}) \\
&\quad \times P(w_{(q+1)e}|N^g_{(q+1)e})] + [\sum_{f,g}\sum_{e=1}^{p-1} P(w_{1(e-1)}, w_{(q+1)m}, N^f_{eq}) \\
&\quad \times P(N^g_{e(p-1)}, N^j_{pq}|N^f_{eq}) P(w_{e(p-1)}|N^g_{e(p-1)})] \\
&= [\sum_{f,g}\sum_{e=q+1}^{m} \alpha_f(p,e) P(N^f \rightarrow N^j\ N^g) \beta_g(q+1,e)] \\
&\quad + [\sum_{f,g}\sum_{e=1}^{p-1} \alpha_f(e,q) P(N^f \rightarrow N^g\ N^j) \beta_g(e,p-1)]
\end{aligned}$$

4, Inside-outside algorithm for learning

Parameter learning

– Assume a fixed set of grammar rules and try to learn their probabilities $\Theta^* = \arg\max_\Theta P(X|\Theta)$

Expectation-maximization (EM)

– E-step: compute the expected counts

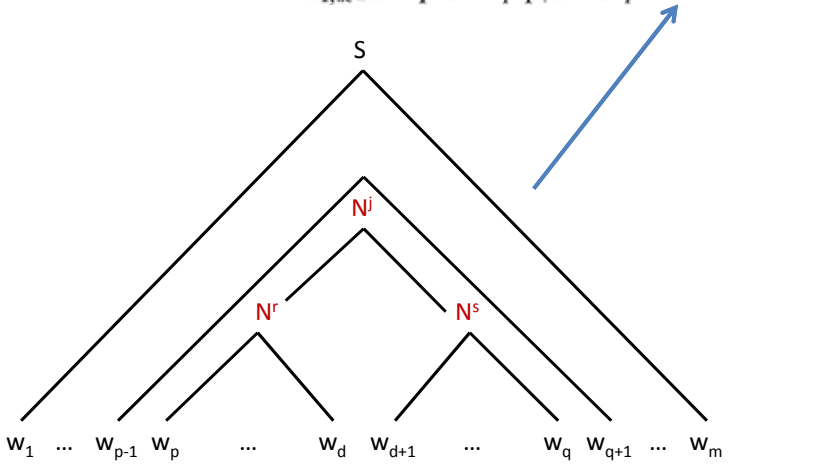
$$C(N^j \to N^r N^s \text{ used}|X, \Theta^t)$$

– M-step: update the probabilities

$$\theta_{jrs}^{t+1} = P(N^j \to N^r N^s) = \frac{C(N^j \to N^r N^s \text{ used}|X,\Theta^t)}{C(N^j \text{ used}|X,\Theta^t)}$$

Expected counts

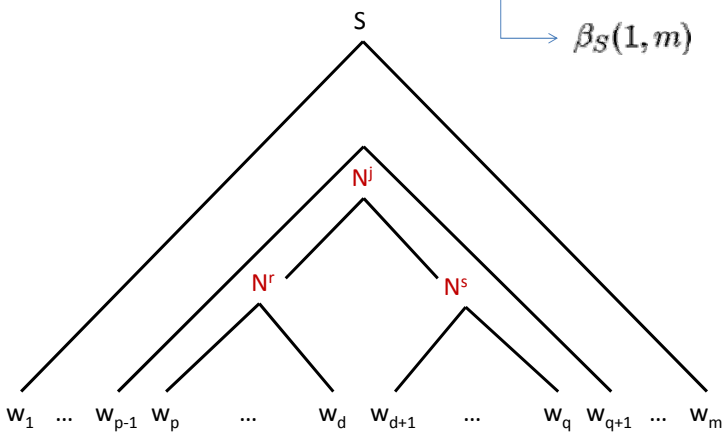
$$C(N^j \to N^r N^s \text{ used}|X, \Theta^t) = \sum_{w_{1,m} \in X} \sum_{p=1}^{m-1} \sum_{q=p+1}^{m} \sum_{d=p}^{q-1} P(\ldots|w_{1,m}, \Theta^t)$$



Expected counts

$$P(\ldots|w_{1,m},\Theta^t) = \frac{\alpha_j(p,q)P(N^j \to N^r N^s)\beta_r(p,d)\beta_s(d+1,q)}{P(w_{1,m})}$$

$$P(w_{1,m}) \rightarrow \beta_S(1,m)$$



Inside-outside algorithm for learning

Initialize the probabilities (e.g., random)

Repeat until convergence

- E-step
  - Compute the inside probabilities
  - Compute the outside probabilities
  - Compute the expected counts
- M-step
  - Update the probabilities

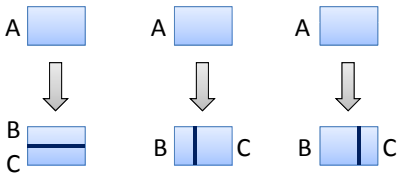
Comments

- Cubic time complexity.
  -- Can be very slow when the data or the grammar is large.
- Converges to local minima.
- Can also be used to optimize the posterior probability

5, Tangram model for images

A variant of PCFG for images

- Each terminal is an image patch.
- Each nonterminal corresponds to a rectangle area of the image
  - The start symbol corresponds to the whole image.
- Each production rule A→BC splits the rectangle area of A either vertically or horizontally into the rectangle areas of B and C

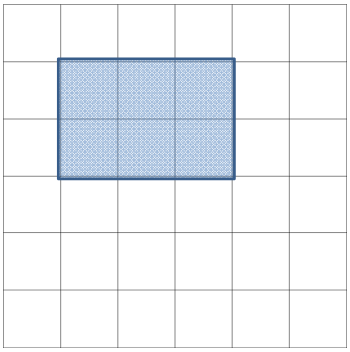


Inside probabilities

The probability of the non-terminal N^j generating a specific rectangle area of the image

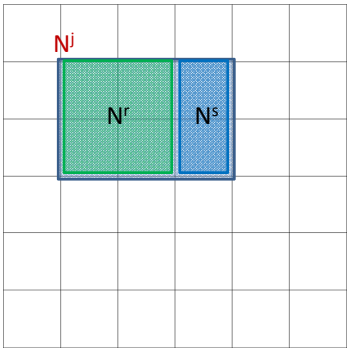
We quantize the image lattice with finite domain.

N^j 



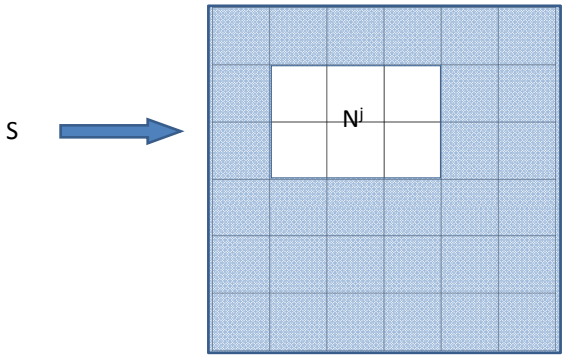
Computing inside probabilities

Bottom-up recursion



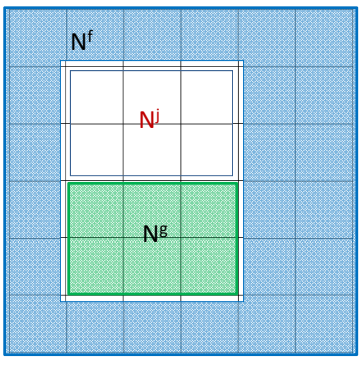
Outside probabilities

The probability of the grammar generating the whole image except a specific rectangle area where the nontermial N^j is generated.



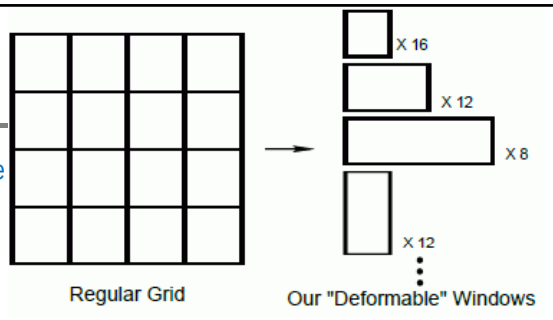
Computing outside probabilities

Top-down recursion



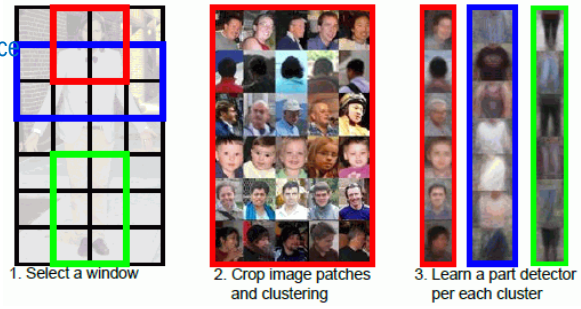
Learning terminal notes in grammar

1, Quantizing the geometric space
--- search all rectangular windows in a 4x4 grid.



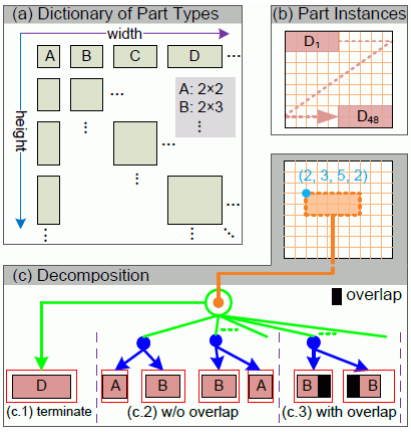
2, Quantizing the appearance space
--- EM clustering for each window
HoG or HIT

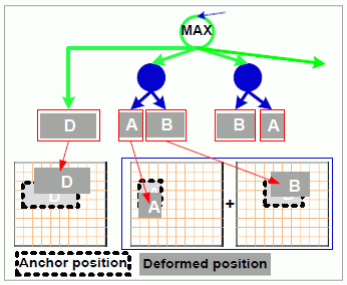
Jungseock Joo (unpublished)



Learning nodes in and-or grammar

Exhaustive enumeration of all rectangle windows in a grid, and binary split of and-node.
local deformation is allowed.





Discriminatively trained AOT
X. Song, TF. Wu et al CVPR 2103.

Learning and-or grammar

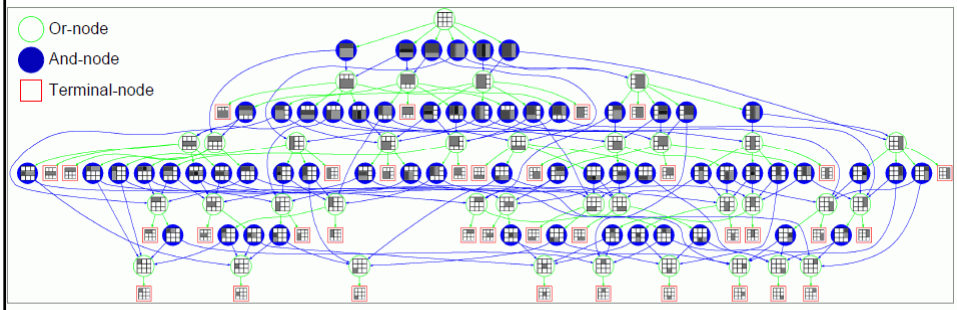


Figure 4. Illustration of the directed acyclic And-Or Graph (AOG) proposed to explore the space of latent structures of objects in this paper. For clarity, we show the AOG structure constructed for unfolding part configurations in a 3×3 grid. The AOG can generate all possible part configurations (the number is often huge for typical grid sizes, see Tabel.1), while allowing efficient exploration with a DP algorithm due to the property of being directed acyclic. See text for details. (Best viewed in color and magnification)

Grid	min. part	#Config.	#Term.	#And
3×3	1×1	319	35	48
5×5	1×1	76,879,359	224	600
10×12	2×2	3.8936e+009	1409	5209

Table 1. The number of part configurations generated from our AOG without considering the overlapped compositions.

Learning and-or grammar

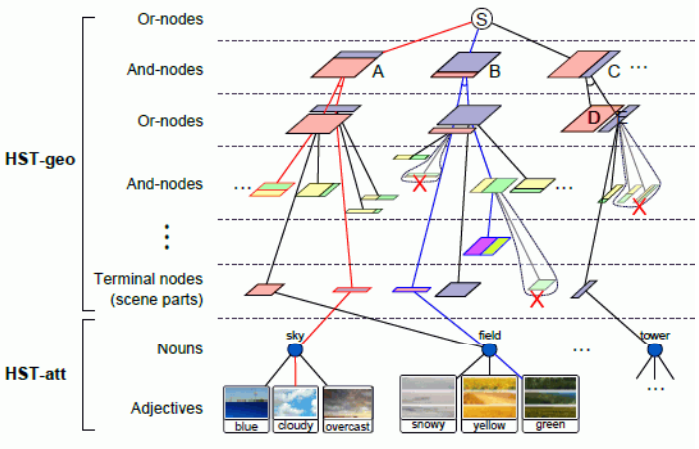
Performance on PASCAL VOC 2007 20 object class.

Table 2. Performance comparison using Average Precision (AP) for the 20 object categories in PASCAL VOC2007 dataset (using the protocol, competition "comp3" trained on VOC2007). All the 5 models use the HOG feature only, and the performance are obtained without post-processing such as bounding box prediction or layout context rescoring. We obtain better performance for 17 object classes.

	aero	bike	boat	bttle	bus	car	mbik	train	bird	cat	cow	dog	hrse	sheep	pers	plant	chair	tble	sofa	tv	avg.
DPM [10]	29	54.6	13.4	26.2	39.4	46.4	37.8	34	0.6	16.1	16.5	5	43.6	17.3	35	8.8	16.3	24.5	21.6	39	26.3
voc-r4 [9]	29.6	57.3	**17.1**	25.2	47.8	55	46.5	44.5	10.1	18.4	24.7	11.2	57.6	18.6	42.1	12.2	21.6	23.3	31.9	40.9	31.8
voc-r5 [12]	32.4	57.7	15.7	25.3	51.3	54.2	47.5	44.2	10.7	17.9	24	11.6	55.6	**22.6**	**43.5**	14.5	21	25.7	34.2	41.3	32.5
3-layer [27]	29.4	55.8	14.3	28.6	44	51.3	38.4	36.8	9.4	21.3	19.3	12.5	50.4	19.7	36.6	15.1	20	25.2	25.1	39.3	29.6
Ours	**35.3**	**60.2**	16.6	**29.5**	**53**	**57.1**	**49.9**	**48.5**	**11**	**23**	**27.7**	**13.1**	**58.9**	22.4	41.4	**16**	**22.9**	**28.6**	**37.2**	**42.4**	**34.7**

Discriminatively trained AOT
X. Song, TF. Wu et al CVPR 2103.

Tangram model and hierarchical spatial tiling



By J. Zhu, T.F. Wu, 2011-2012;
S. Wang, J. Joo 2012-2013

Tangram model and hierarchical spatial tiling

Scene parsing and attribute tagging with the learning AOG (S. Wang et al CVPR 2013.)