Supplementary Material for Generalized Earley Parser

Siyuan Qi [1] **Baoxiong Jia** [1 2] **Song-Chun Zhu** [1]

1. Review of Formal Grammar

Figure 1 shows an example of English grammar. We review some definitions in the formal language theory:

- Grammar: a set of rules by which valid sentences in a language are constructed.
- Non-terminal: a grammar symbol that can be replaced/expanded to a sequence of symbols.
- Terminal: an actual word in a language; these are the symbols in a grammar that cannot be replaced by anything else. ”terminal” is supposed to conjure up the idea that it is a dead-end: pno further expansion is possible.
- Production: a grammar rule that describes how to replace/exchange symbols.

Parsing is the process of analyzing a string of symbols, conforming to the rules of a formal grammar. Figure 2 shows the function of a parsing algorithm: it analyzes a sentence and generates a parse tree.

<sentence>	→	<subject> <verbphrase> <object>
<subject>	→	“This” | “Computers” | “I”
<verbphrase>	→	<adverb> <verb> | <verb>
<adverb>	→	“never”
<verb>	→	“is” | “run” | “am” | “tell”
<object>	→	“the” noun | “a” noun | noun
noun	→	“university” | “world” | “lies”

Figure 1. Production rules of an example English grammar. “|” means “or”. Non-terminal symbols are denoted by $< \cdot >$, and terminal symbols are denoted by “ · ”. Example sentences from the language defined by this grammar are: “This is a university”, “Computers run the world”, “I never tell lies”.

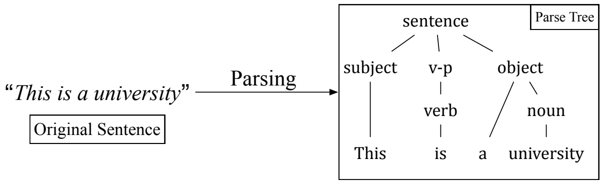

Figure 2. Parsing a sentence according to grammar productions.

2. Example of Generalized Earley Parser

Figure 3 shows a walk-through example of running the generalized Earley parser, which corresponds to Figure 3 in the original paper.. The given language contains three terminal symbols: “0”, “1”, and “+”. The input is a 5×3 matrix, containing the classification confidence of 3 symbols for 5 frames. The cached probabilities for different prefixes are computed along the inference process. The algorithm essentially expands a prefix tree by three operations: scanning, prediction, and completion. The final parsing output is the optimal string “$0 + 1$” with probability 0.43.

Sample grammar: $\gamma \rightarrow R$; $R \rightarrow R + R$; $R \rightarrow 0|1$

Input (classifier output)

Frame	0	1	+
0	0.8	0.1	0.1
1	0.8	0.1	0.1
2	0.1	0.1	0.8
3	0.1	0.8	0.1
4	0.1	0.8	0.1

Cached probability

Frame	ϵ	0	1	0 +	0 + 0	0 + 1	0 + 1 +
0	0	8e-1	0.1	0	0	0	0
1	0	6.4e-1	1e-2	8e-2	0	0	0
2	0	6.4e-2	1e-3	0.58	8e-3	8e-3	0
3	0	6.4e-3	8e-4	6.4e-2	5.8e-2	0.47	8e-4
4	0	6.4e-4	6.4e-5	7e-4	1.2e-2	0.42	4.7e-2
prefix	1	8e-1	0.1	0.60	7.2e-2	0.52	4.8e-3

State: | state # | rule | origin | prefx | comment |

$S(0,0) : l = \epsilon, p(l) = 0.0, p(l...) = 1.0$

(1)	$\gamma \rightarrow \cdot R$	0, 0	ϵ	start rule
(2)	$R \rightarrow \cdot R + R$	0, 0	ϵ	predict: (1)
(3)	$R \rightarrow \cdot 0$	0, 0	ϵ	predict: (1)
(4)	$R \rightarrow \cdot 1$	0, 0	ϵ	predict: (1)

$S(1,0) : l = “0”, p(l) = 6.4e-4, p(l...) = 0.8$

(1)	$R \rightarrow 0 \cdot$	0, 0	“0”	scan: $S(0,0)(3)$
(2)	$R \rightarrow R \cdot +R$	0, 0	“0”	complete: (1) and $S(0,0)(2)$
(3)	$\gamma \rightarrow R \cdot$	0, 0	“0”	complete: (2) and $S(0,0)(1)$

$S(1,1) : l = “1”, p(l) = 6.4e-4, p(l...) = 0.1$

(1)	$R \rightarrow 0 \cdot$	0, 0	“1”	scan: $S(0,0)(4)$

$S(2,0) : l = “0+”, p(l) = 7.0e-3, p(l...) = 0.599$

(1)	$R \rightarrow R + \cdot R$	0, 0	“$0+$”	scan: $S(1,0)(2)$
(2)	$R \rightarrow \cdot R + R$	2, 0	“$0+$”	predict: (1)
(3)	$R \rightarrow \cdot 0$	2, 0	“$0+$”	predict: (1)
(4)	$R \rightarrow \cdot 1$	2, 0	“$0+$”	predict: (1)

$S(3,0) : l = “0+0”, p(l) = 1.2e-2, p(l...) = 7.2e-2$

(1)	$R \rightarrow 0 \cdot$	2, 0	“$0+0$”	scan: $S(2,0)(3)$

$S(3,1) : l = “0+1”$, **p(l)=0.43**, $p(l...) = 0.52$

(1)	$R \rightarrow 1 \cdot$	2, 0	“$0+1$”	scan: $S(2,0)(4)$
(2)	$R \rightarrow R + R \cdot$	0, 0	“$0+1$”	complete: (1) and $S(2,0)(1)$
(3)	$R \rightarrow R \cdot +R$	2, 0	“$0+1$”	complete: (1) and $S(2,0)(2)$
(4)	$\gamma \rightarrow R \cdot$	0, 0	“$0+1$”	complete: (2) and $S(0,0)(1)$

$S(4,0) : l = “0+1+”, p(l) = 4.7e-2, p(l...) = 4.8e-2$

(1)	$R \rightarrow 0 \cdot$	2, 0	“$0+0$”	scan: $S(3,1)(3)$

Final output: $l^* =$ “$0+1$” with probability 0.43

Figure 3. An example of the generalized Earley parser. This example corresponds to Figure 3 in the original paper.

[1] University of California, Los Angeles, USA [2] Peking University, Beijing, China. Correspondence to: Siyuan Qi <syqi@cs.ucla.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

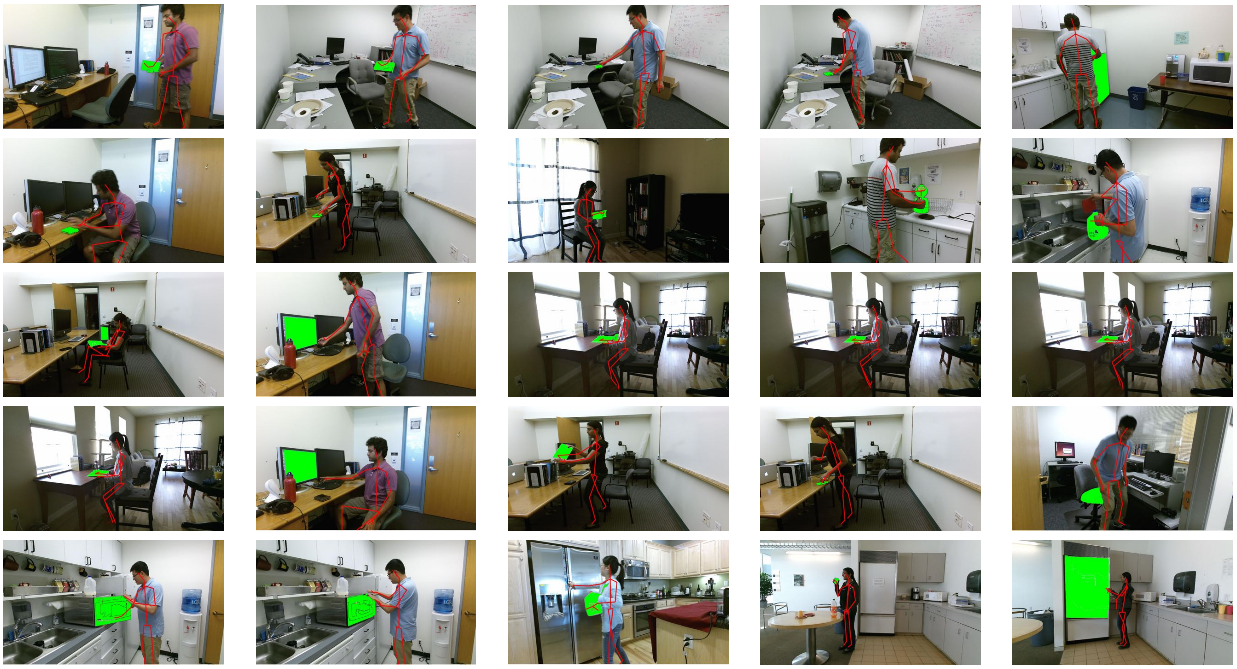

Figure 4. Visualization of the selected super-pixels for extracting kernel descriptors

3. Feature Extraction

For the CAD-120 dataset, we use the publicly available features from KGS (Koppula et al., 2013). For the Watch-n-Patch dataset, we extract the same features as described in (Wu et al., 2015) for all methods. The features are composed of skeleton features and human-object interaction features. The skeleton features include angles between the connected parts, the change of joint positions and angles regarding previous frames and the first frame. We used zero values for those frames in which the joints are not tracked. The interaction object features are extracted based on RGB-D images. First of all, we calculate the super-pixel segmentation for both RGB and depth images. For each frame, an edge detection algorithm (Dollár & Zitnick, 2013) is applied to both the resized RGB (1920×1080 to 960×540) and the depth images. The selected super-pixels for are shown in Figure 2. Next, we extract super-pixel segmentations from the contour maps generated by the edge detection algorithm. In this case, 0.05 and 0.13 are chosen as the segmenting threshold values for RGB and depth respectively. Second, a moving object detection algorithm is used for monitoring the foreground mask in each frame. Finally, we extract features from the image segments with more than 50% in the foreground mask and within a distance to the human hand joints in both 3D points and 2D pixels. To model the interactive object features more accurately, we removed the false positive segments which overlap with human joints. Overall, we extracted six kernel descriptors (Wu et al., 2014) from these image segments: gradient, color, local binary pattern, depth gradient, spin, surface normals, and KPCA/self-similarity. We first set grid size to 8 and patch size to 16 for generating feature vectors. Then, we construct kernel descriptors by projecting these feature vectors to the visual words generated by K-means algorithm. In this experiment, we generated visual words separately for office scene and kitchen scene. We used 400 visual words for RGB images and 200 for depth images accordingly.

4. Segmentation Results

The segmentation results on the Watch-n-Patch are shown in Figure 5. Here we visualize the video segmentation of the ground truth labels, the results generated by ST-AOG (Qi et al., 2017), the Bi-LSTM algorithm, and our algorithm.

References

Dollár, P. and Zitnick, C. L. Structured forests for fast edge detection. In *ICCV*, 2013.

Koppula, H. S., Gupta, R., and Saxena, A. Learning human activities and object affordances from rgb-d videos. *IJRR*, 2013.

Qi, S., Huang, S., Wei, P., and Zhu, S.-C. Predicting human activities using stochastic grammar. In *ICCV*, 2017.

Wu, C., Lenz, I., and Saxena, A. Hierarchical semantic labeling for task-relevant rgb-d perception. In *RSS*, 2014.

Wu, C., Zhang, J., Savarese, S., and Saxena, A. Watch-n-patch: Unsupervised understanding of actions and relations. In *CVPR*, 2015.

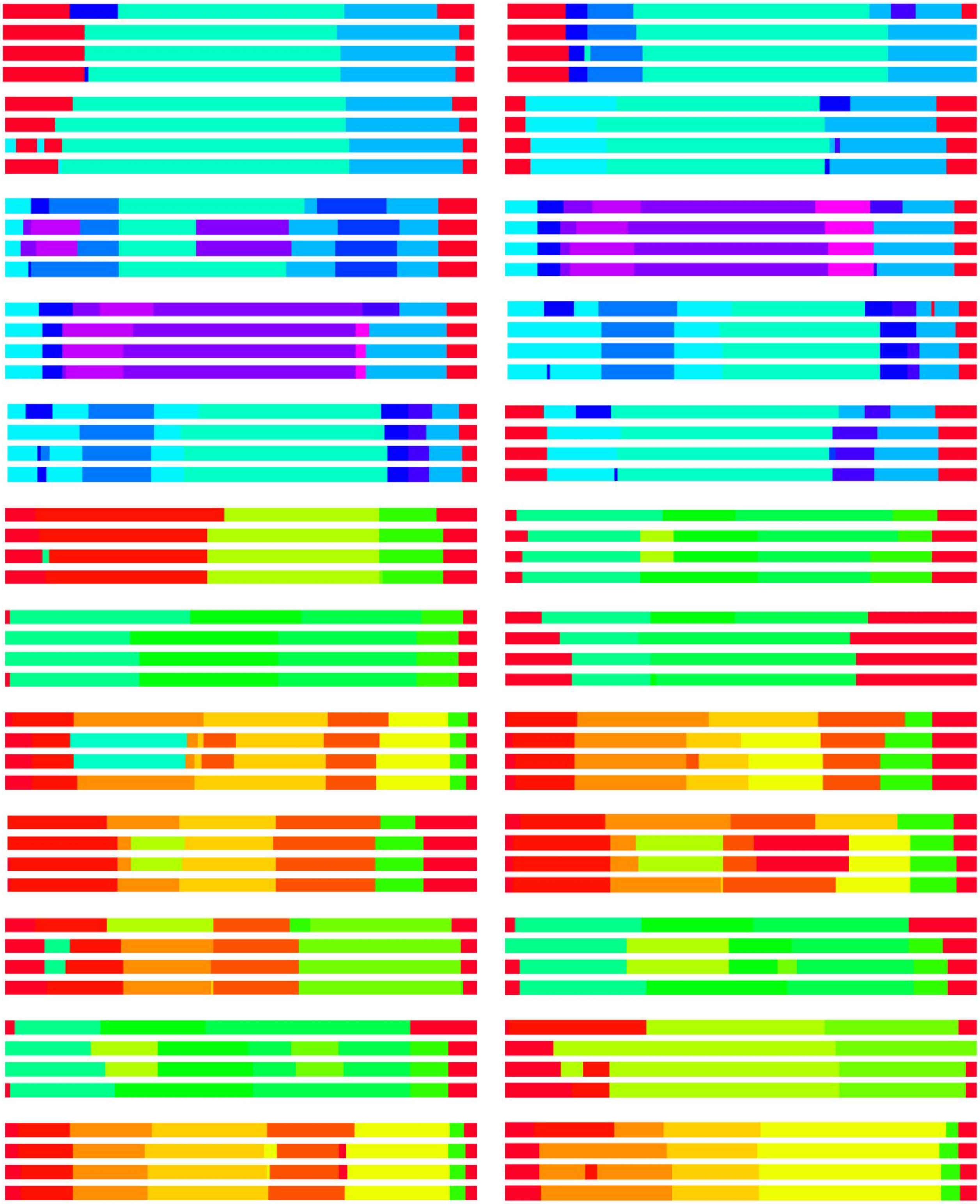

Figure 5. Qualitative results of segmentation results. In each group of four segmentations, the rows from the top to the bottom shows: 1) ground-truth, 2) results of ST-AOG, 3) Bi-LSTM, and 4) Bi-LSTM + generalized Earley parser.