Parsing Language Structures and Meanings

xiang.zhang@nlpr.ia.ac.cn
July 3, 2025

Agenda

- Tutorial: Chart Parsing
- Unsupervised Grammar Induction
- Migration to Semantic Parsing
- What's Next

Agenda

- Tutorial: Chart Parsing
- Unsupervised Grammar Induction
- Migration to Semantic Parsing
- What's Next

Chart Parser Tutorial

- A recogniser determines if a string belongs to a grammar.
  - <----------- Parsing ---------->
  - <--- Recognition ---><------>

- Earley Recogniser

- CYK Recogniser

- The Semi-ring Parsing and Inside Algorithm

Complete Earley Parser

```
Sum -> Sum [+-] Product
Sum -> Product
Product -> Product [*/] Factor
Product -> Factor
Factor -> '(' Sum ')'
Factor -> Number
Number -> [0-9] Number
Number -> [0-9]
```

1	+	(	2	*	3	-	4	)

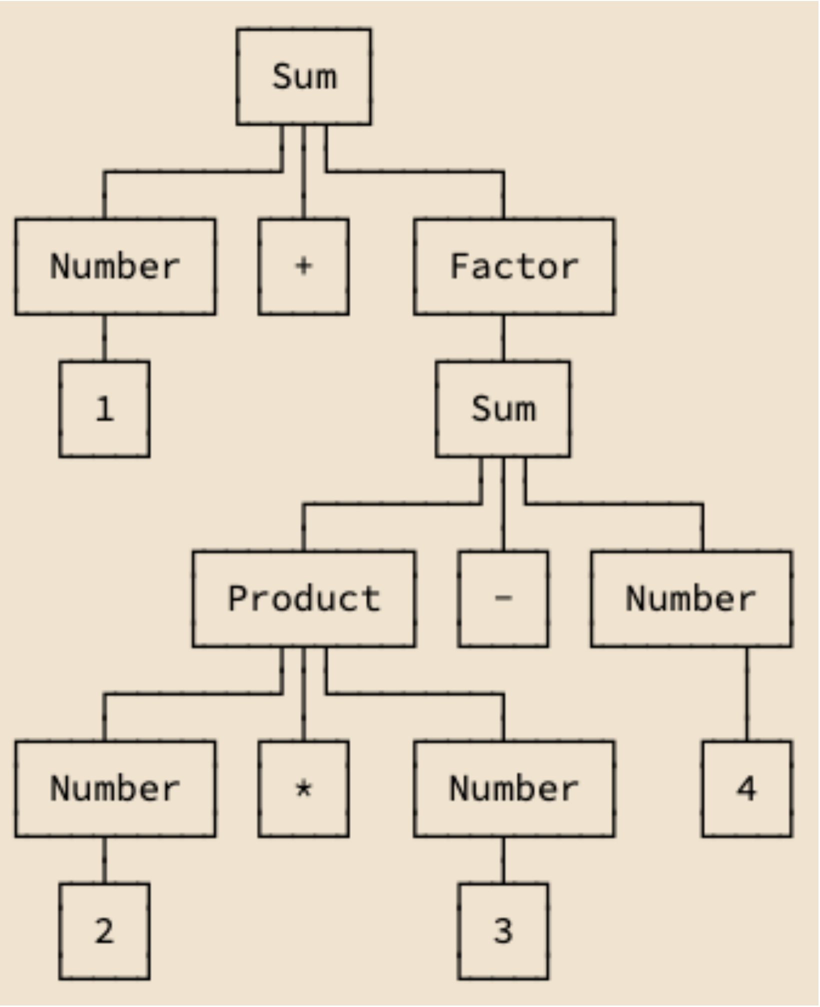
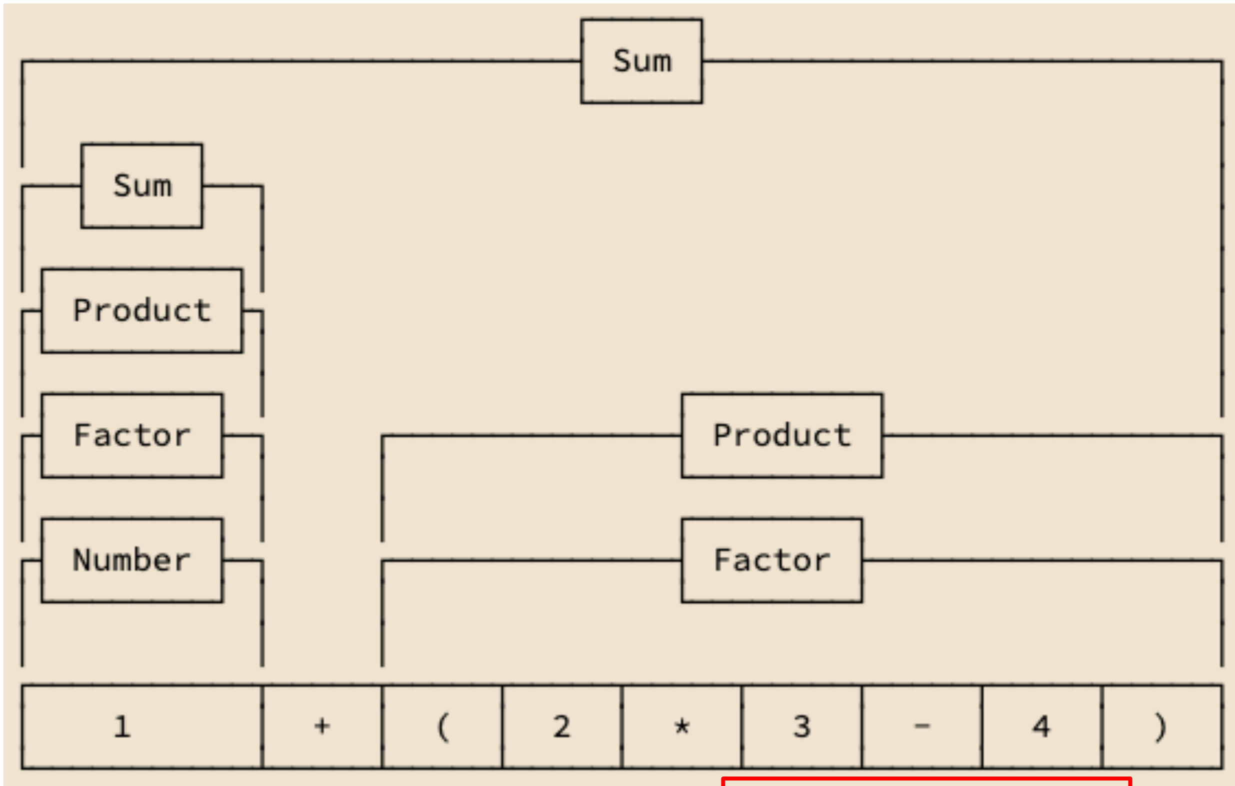
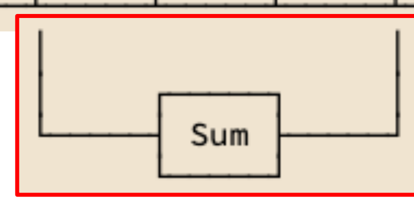


Chart Parser (Partial Parses)

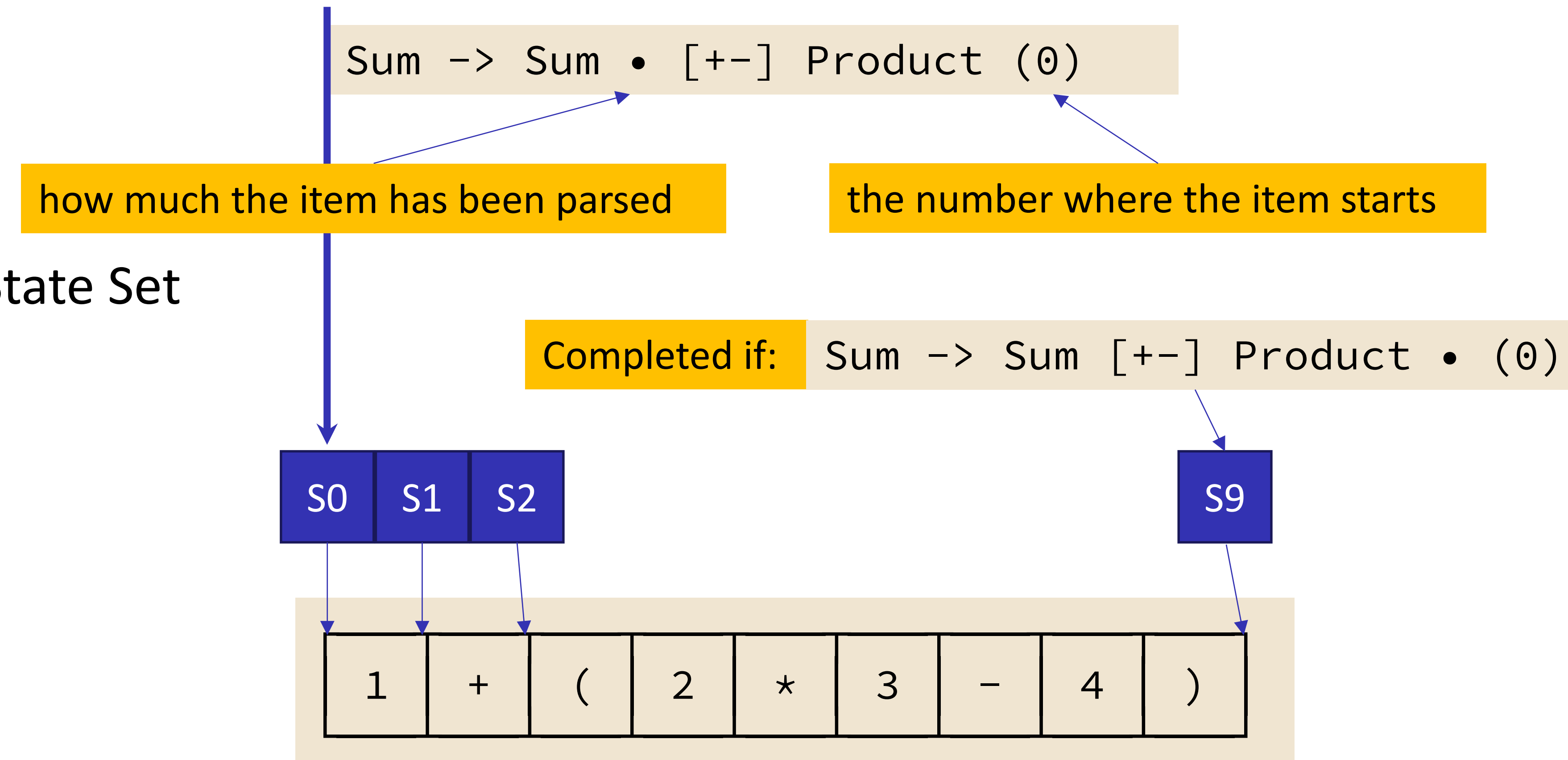




avoid most unnecessary work by not even trying a whole slew of hopeless partial parses like this one

Earley Recognition (1/3)

- Earley Item



- State Set

Earley Recognition (2/3)

- for s in state set, for item in s
  - Prediction: add rule of the next non-terminal to the current set
  - Scan: move fat dot forward, add this item into the next set.

initialized with
- start rule,
- (0),
- leading fat dot

S0

```
Sum -> • Sum [+-] Product (0)
Sum -> • Product          (0)
```

predicted

```
Product -> • Product [*/] Factor  (0)
Product -> • Factor               (0)
```

S1

```
Number -> [0-9] • Number (0)
```

scanned

1	+	(	2	*	3	-	4	)

Earley Recognition (3/3)

- completion when the dot is at the end

```
Number  -> [0-9] •                (0)
```

S0

```
Sum     -> • Sum     [+-] Product (0)
Sum     -> • Product              (0)
Product -> • Product [*/] Factor  (0)
Product -> • Factor               (0)
Factor  -> • '(' Sum ')'          (0)
Factor  -> • Number               (0)
Number  -> • [0-9] Number         (0)
Number  -> • [0-9]                (0)
```

S1

```
Factor  -> Number •               (0)
```

CYK Recognition Illust.

- The Cocke–Younger–Kasami algorithm for ambiguous strings
  - bottom-up parsing
  - Rules are in the Chomsky Norm Form

S → PRP VP
NP → PRP PP
NP → DT NNS
VP → VBD PRP
VP → VBD NP
VP → VP PP
PP → IN NP

l=5						
l=4						
l=3						
l=2						
l=1						
l=0	PRP	VBD NN	PRP	IN	DT	NNS
	I	saw	him	with	the	binoculars

CYK Recognition Illust. (l=1)

l=5						
l=4						
l=3						
l=2						
l=1		VP			NP	
l=0	PRP	VBD NN	PRP	IN	DT	NNS
	I	saw	him	with	the	binoculars

S → PRP VP
NP → PRP PP
NP → DT NNS
VP → VBD PRP
VP → VBD NP
VP → VP PP
PP → IN NP

CYK Recognition Illust. (l=2)

l=5						
l=4						
l=3						
l=2	S			PP		
l=1		VP			NP	
l=0	PRP	VBD NN	PRP	IN	DT	NNS
	I	saw	him	with	the	binoculars

S → PRP VP
NP → PRP PP
NP → DT NNS
VP → VBD PRP
VP → VBD NP
VP → VP PP
PP → IN NP

CYK Recognition Illust. (l=3)

l=5						
l=4						
l=3			NP			
l=2	S			PP		
l=1		VP			NP	
l=0	PRP	VBD NN	PRP	IN	DT	NNS
	I	saw	him	with	the	binoculars

S → PRP VP
NP → PRP PP
NP → DT NNS
VP → VBD PRP
VP → VBD NP
VP → VP PP
PP → IN NP

CYK Recognition Illust. (l=4)

l=5						
l=4		VP/VP				
l=3			NP			
l=2	S			PP		
l=1		VP			NP	
l=0	PRP	VBD NN	PRP	IN	DT	NNS
	I	saw	him	with	the	binoculars

S → PRP VP
NP → PRP PP
NP → DT NNS
VP → VBD PRP
VP → VBD NP
VP → VP PP
PP → IN NP

CYK Recognition Illust. (l=5)

l=5	S					
l=4		VP				
l=3			NP			
l=2	S			PP		
l=1		VP			NP	
l=0	PRP	VBD NN	PRP	IN	DT	NNS
	I	saw	him	with	the	binoculars

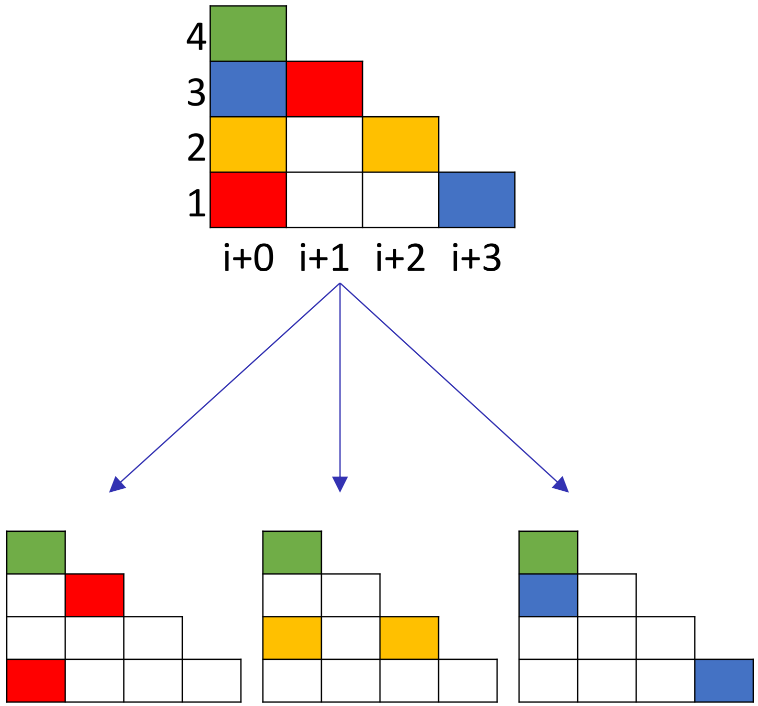
S → PRP VP
NP → PRP PP
NP → DT NNS
VP → VBD PRP
VP → VBD NP
VP → VP PP
PP → IN NP

CYK Recognition Code

```
chart[1..n, 1..n, 1..V] = False
for p = 1 to n:
 for rule A->w_p in rules:
    chart[1, p, A] := True

for l = 2 .. n:
 for p = 1 .. n - l + 1:
    for s = 1 .. l - 1:
      for rule A->BC :
         chart[l, p, A] =
           chart [l, p, A] or
           chart[s, p, B] and
           chart[l-s, p+s, c]

return chart[n, 1, S]
```



The ||| cat is jumping
The cat ||| is jumping
The cat is ||| jumping

Semiring-based Parsing

- Semi-ring: (A, $\oplus$, $\otimes$, 0, 1)
  - $\oplus$: commutative
  - $\otimes$: associative
  - CYK recognition is ({True, False}, OR, AND, True, False)

- Inside Algorithm
  - ($\mathbb{R}_{\geqslant 0} \cup \{+\infty\}$, +, *, 0, 1)
  - for all marginal tree weights

- State Transition Equation:

$$A_i^k = \sum_B \sum_C \sum_j \pi_{A \to BC} B_i^j C_{j+1}^k$$

Agenda

- Tutorial: Chart Parsing
- Unsupervised Grammar Induction
- Migration to Semantic Parsing
- What's Next

Grammar Induction

- What do we mean when we say "grammar"?
  - Conventional Rules, with less observation
  - A mixed idea of language without pragmatics in the ESL education
  - Explain the human language ability as suggested by generative grammar

- Learning grammar rules from data
  - Usually, the grammar formalization is assumed beforehand

- Selected Works
  - DIORA (NAACL 2019)
  - C-PCFG (ACL 2019) and TD-PCFG (NAACL 2021)
  - Perturb-and-Parse (ACL 2019)
  - R2D2 (ACL 2021)

Techniques for Grammar Induction

Family	Method	Structure or Not	Discrete Output	Inference Algo.
Surrogate	STE (Hinton, 2012)	Both	✓	MAP
	SPIGOT (Peng et al., 2018)	✓	✓	MAP
Relaxation	softmax	✗	✗	
	sparsemax (Martins and Astudillo, 2016) +[†]	✗	✓	
	Part-Marginalization	✓	✗	Marg.
	SparseMAP (Niculae et al., 2018a)	✓	✓	MAP
Sampling	Score Function Estimator (Williams, 1992)	Both	✓	Sampling
	Rectified Distributions (Louizos et al., 2018)	✗	✗[‡]	
	Gumbel-Max (Gumbel, 1954)	✗	✓	
	Gumbel-softmax (Jang et al., 2017; Maddison et al., 2017)	✗	✗	
	Perturb-and-Parse (Corro and Titov, 2019a)	✓	✗	MAP
	Direct Loss Minimization (McAllester et al., 2010)	✗	✓	

DIORA
C-PCFG
TD-PCFG
R2D2

Perturb-and-Parse

Zhaofeng Wu. 2022. Learning with Latent Structures in Natural Language Processing: A Survey. *arXiv:2201.00490*

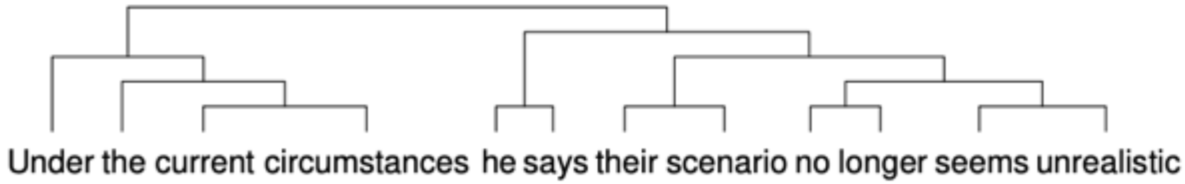
Unsupervised Latent Tree Induction with Deep Inside-Outside Recursive Autoencoders
NAACL 2019

Andrew Drozdov*, **Pat Verga***, **Mohit Yadav***,
Mohit Iyyer, and **Andrew McCallum**

College of Information and Computer Sciences
University of Massachusetts Amherst

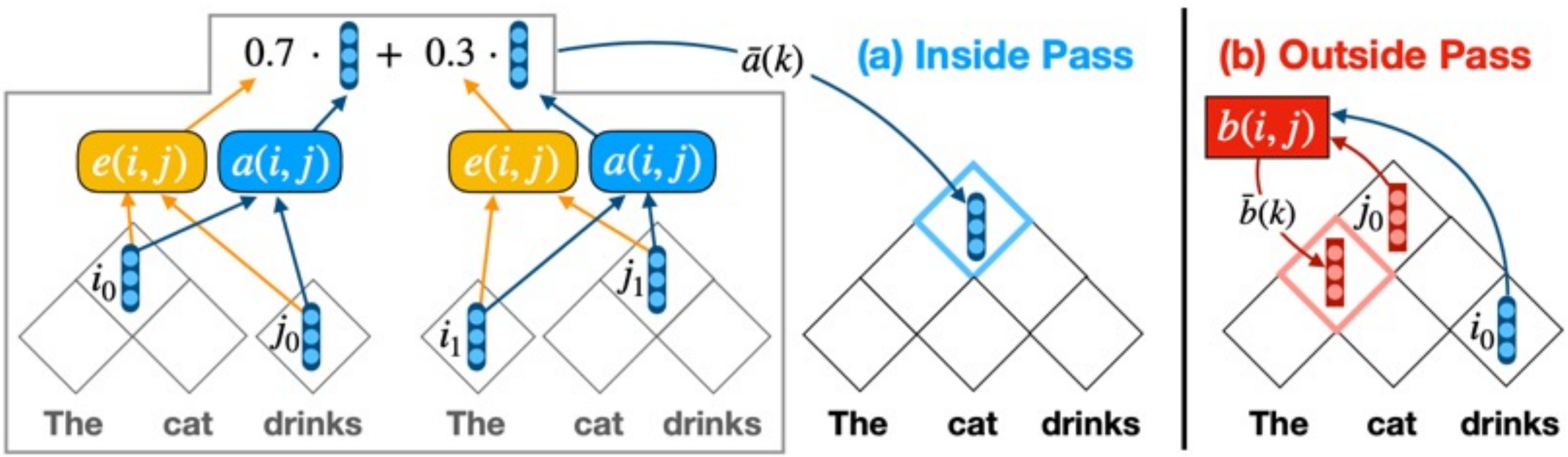
DIORA: Motivations

- Classical parsers
  - require annotated treebanks limited in size and domain
- Latent tree parsers
  - produce representations for all internal nodes
  - each generated with a soft weighting over all possible sub-trees
  - requires sentence level annotations for training (usually labels for downstream tasks, such as NLI)
- Previous works
  - predict trees not aligned with known treebanks
  - no mechanism to model phrases, requiring a complex procedure to extract syntactic structures (such as ON-LSTM)



DIORA: Method

- Procedure Illustration



- Losses:

$$L_{\boldsymbol{x}} = \sum_{i=0}^{T-1} \sum_{i^*=0}^{N-1} \max(0, 1 - \bar{b}(i) \cdot \bar{a}(i) + \bar{b}(i) \cdot \bar{a}(i^*))$$

$$Z^* = \sum_{i^*=0}^{N-1} \exp(\bar{b}(i) \cdot \bar{a}(i^*))$$

$$L_{\boldsymbol{x}} = -\sum_{i=0}^{T-1} \log \frac{\exp(\bar{b}(i) \cdot \bar{a}(i))}{\exp(\bar{b}(i) \cdot \bar{a}(i)) + Z^*}$$

Compound Probabilistic Context-Free Grammars for Grammar Induction

ACL 2019

Yoon Kim
Harvard University
Cambridge, MA, USA
yoonkim@seas.harvard.edu

Chris Dyer
DeepMind
London, UK
cdyer@google.com

Alexander M. Rush
Harvard University
Cambridge, MA, USA
srush@seas.harvard.edu

C-PCFG: Motivations

- Direct methods are found difficult to induce PCFG from data
  - Ill-behaved optimization landscape
  - Overly strict independence assumptions of PCFGs
- Successful approaches resort to
  - carefully-crafted auxiliary objectives
  - priors or non-parametric models
  - manually engineered features
- They propose to
  - parameterizing PCFG with neural networks makes it possible to induce linguistically meaningful grammars by simply optimizing log-likelihood
  - to incorporate side information is straight-forward

C-PCFG: Models (1)

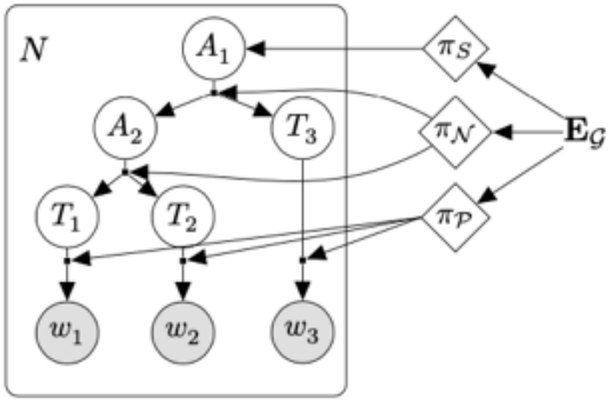
- Neural Parameterization (N-PCFG)

$$
\begin{aligned}
S &\to A, && A \in \mathcal{N} \\
A &\to B\,C, && A \in \mathcal{N},\ B, C \in \mathcal{N} \cup \mathcal{P} \\
T &\to w, && T \in \mathcal{P},\ w \in \Sigma.
\end{aligned}
$$

$$
\pi_{S \to A} = \frac{\exp(\mathbf{u}_A^\top f_1(\mathbf{w}_S))}{\sum_{A' \in \mathcal{N}} \exp(\mathbf{u}_{A'}^\top f_1(\mathbf{w}_S))},
$$

$$
\pi_{A \to BC} = \frac{\exp(\mathbf{u}_{BC}^\top \mathbf{w}_A)}{\sum_{B'C' \in \mathcal{M}} \exp(\mathbf{u}_{B'C'}^\top \mathbf{w}_A)},
$$

$$
\pi_{T \to w} = \frac{\exp(\mathbf{u}_w^\top f_2(\mathbf{w}_T))}{\sum_{w' \in \Sigma} \exp(\mathbf{u}_{w'}^\top f_2(\mathbf{w}_T))},
$$

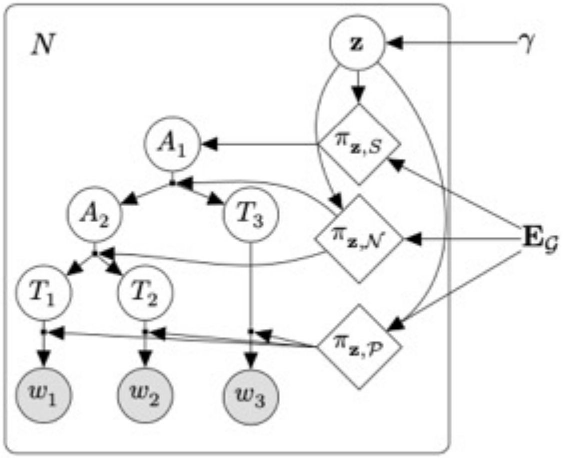


C-PCFG: Models (2)

- Compound PCFGs (C-PCFG)
  - for grammar induction, first-order context-free assumption is adopted not because its adequacy but its tractability.
  - C-PCFG is a restricted version of some higher-order PCFG

$$
\begin{aligned}
S &\to A, & A &\in \mathcal{N} \\
A &\to B\,C, & A &\in \mathcal{N},\ \ B, C \in \mathcal{N} \cup \mathcal{P} \\
T &\to w, & T &\in \mathcal{P},\ \ w \in \Sigma.
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{z} \sim p_\gamma(\mathbf{z}), &\qquad \boldsymbol{\pi}_\mathbf{z} = f_\lambda(\mathbf{z}, \mathbf{E}_\mathcal{G}), \\
\pi_{\mathbf{z}, S \to A} &\propto \exp(\mathbf{u}_A^\top f_1([\mathbf{w}_S; \mathbf{z}])), \\
\pi_{\mathbf{z}, A \to BC} &\propto \exp(\mathbf{u}_{BC}^\top [\mathbf{w}_A; \mathbf{z}]), \\
\pi_{\mathbf{z}, T \to w} &\propto \exp(\mathbf{u}_w^\top f_2([\mathbf{w}_T; \mathbf{z}])),
\end{aligned}
$$



C-PCFG: Training

- For a simple N-PCFG:

$$\log p_\theta(x) = \log \sum_{t \in T_{G(x)}} p_\theta(t)$$

- For the compound PCFG

$$\log p_\theta(x) = \log \left(\int \sum_{t \in T_{G(x)}} p_\theta(t \mid z) p_\gamma(z) dz \right)$$

  - is intractable, resort to a collapsed amortized variational inference instead

$$\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x \mid z)] - \mathrm{KL}[q_\phi(z \mid x) \parallel p_\gamma(z)]$$

- For inference, use the mean vector to approximate z

PCFGs Can Do Better: Inducing Probabilistic Context-Free Grammars with Many Symbols

NAACL 2021

Songlin Yang[♣], Yanpeng Zhao[◇], Kewei Tu[♣*]

[♣]School of Information Science and Technology, ShanghaiTech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences
[◇]ILCC, University of Edinburgh
{yangsl,tukw}@shanghaitech.edu.cn
yannzhao.ed@gmail.com

TD-PCFG: Motivations

- Inside algorithm is cubic computational complexity
  - e.g. C-PCFG uses 30 non-terminals and 60 pre-terminals

- More symbols are important:
  - Dividing PTB categories into subtypes improves parsing
  - Increasing the number of hidden states is helpful for learning latent variables

TD-PCFG: Methods

- Kruskal Decomposition

$$\begin{aligned} T &= \sum_{l=1}^{d} T^{(l)} \\ T_{ijk}^{(l)} &= u_i^{(l)} \cdot v_j^{(l)} \cdot w_k^{(l)} \end{aligned}$$

- Applied to the state-transition equation

$$A_i^k = \sum_B \sum_C \sum_j \pi_{A \to BC} B_i^j C_{j+1}^k$$

  - We have

$$S_{ik} = U \sum_j (V^T S_{ij}) \odot (W^T S_{jk}), \quad U \in \mathbb{R}^{n \times d}, V, W \in \mathbb{R}^{m \times d}$$

  - where U is row-normalized, and V, W are column-normalized

Empirical Results

Model	WSJ Mean	WSJ Max
Left Branching		8.7
Right Branching		39.5
Random Trees	18.1	18.2
Systems without pretrained word embeddings		
PRPN† (Shen et al., 2018a)	47.3	47.9
ON† (Shen et al., 2019)	48.1	50.0
N-PCFG (Kim et al., 2019a)	50.8	52.6
C-PCFG (Kim et al., 2019a)	55.2	60.1
NL-PCFG (Zhu et al., 2020)	55.3	
N-PCFG⋆	$50.9_{\pm 2.3}$	54.6
N-PCFG⋆ w/ MBR	$52.3_{\pm 2.3}$	55.8
C-PCFG⋆	$55.4_{\pm 2.2}$	59.0
C-PCFG⋆ w/ MBR	$56.3_{\pm 2.1}$	60.0
TN-PCFG p = 60 (ours)	$51.4_{\pm 4.0}$	55.6
TN-PCFG p = 500 (ours)	$\mathbf{57.7}_{\pm 4.2}$	**61.4**
Systems with pretrained word embeddings		
DIORA (Drozdov et al., 2019)		56.8
S-DIORA (Drozdov et al., 2020)	57.6	64.0
CT (Cao et al., 2020)	62.8	65.9
Oracle Trees		84.3

Table 1: Unlabeled sentence-level F1 scores on the WSJ test data. † indicates numbers reported by Kim et al. (2019a). ⋆ indicates our reimplementations of N-PCFGs and C-PCFGs. p denotes the preterminal number.

Learning Latent Trees with Stochastic Perturbations and Differentiable Dynamic Programming

ACL 2019

Differentiable Perturb-and-Parse: Semi-Supervised Parsing with A Structured Variational Autoencoder

ICLR 2019

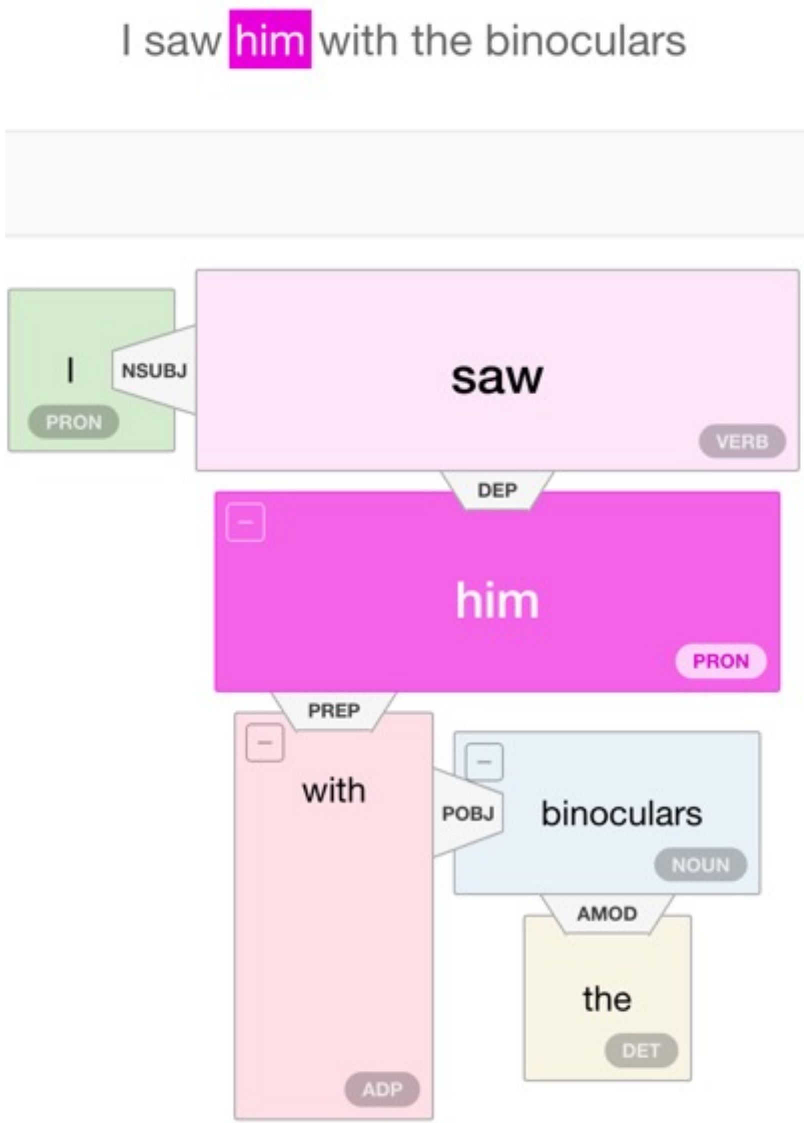
Caio Corro **Ivan Titov**

ILCC, School of Informatics, University of Edinburgh

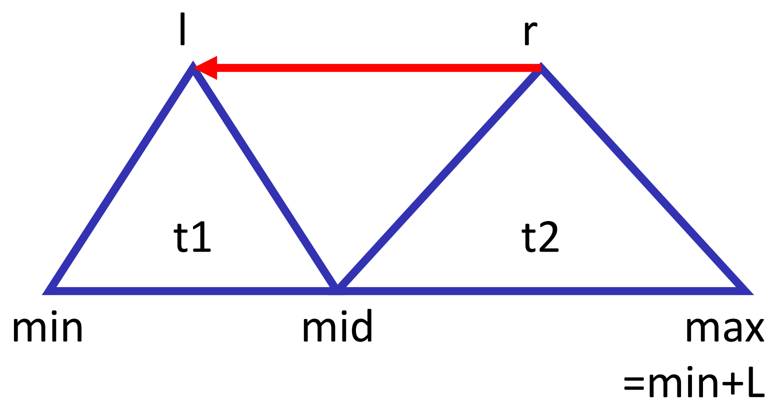
ILLC, University of Amsterdam

c.f.corro@uva.nl ititov@inf.ed.ac.uk

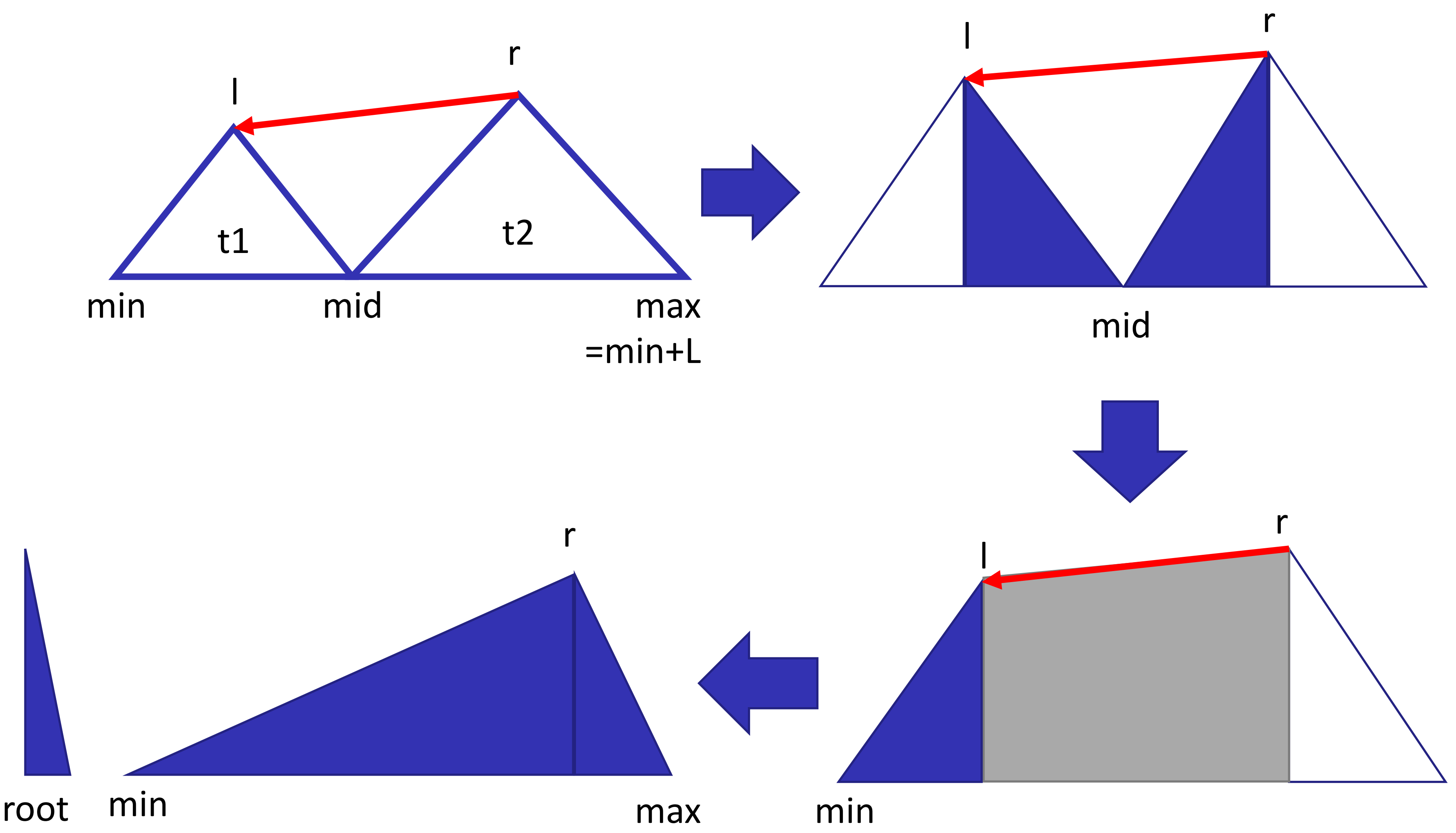
Dependency Parsing



- Collins' algorithm
  - Space: $O(N^3)$
  - Time: $O(N^5)$

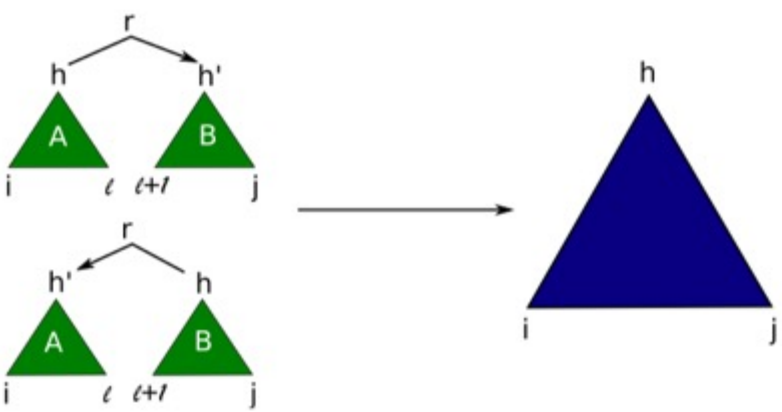


Eisner's Algorithm Illust.

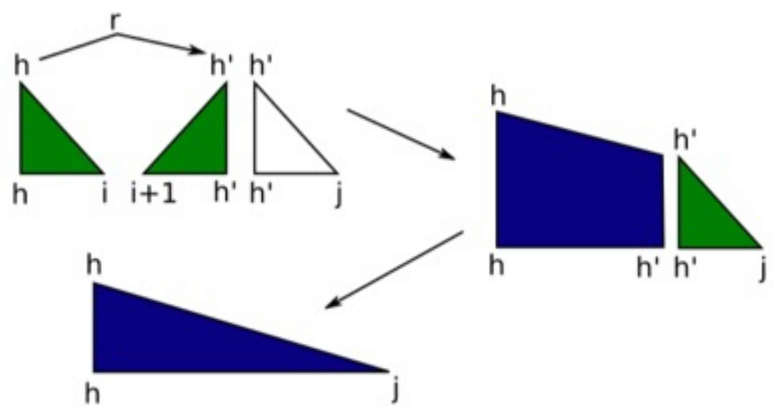


Comparison

- Collins' Algorithm
- Space: $O(N^3)$
- Time: $O(N^5)$
- Chart: [min, max, head]



- Eisner's Algorithm
- Space: $O(N^2)$
- Time: $O(N^3)$
- Chart: [min, max, dir, comp]



Perturb-and-Parse: Motivations

- Previous work on discrete structures
  - require treebank annotations limited in size and domain

- Linguistic structures trained for downstream tasks
  - provide an inductive bias specifying structures
  - not making any assumptions regarding what the structures represent

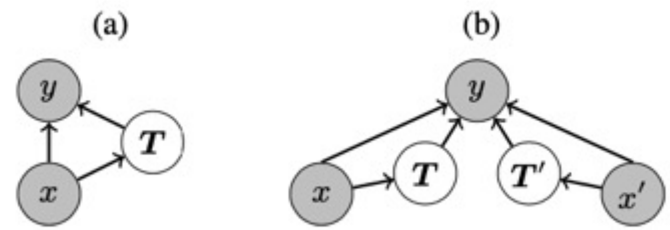
- sample global structures in a differentiable way

Perturb-and-Parse: Methods

- Tree distribution models

$$W_{h,m} = \mathrm{MLP}^{\mathrm{head}}(\boldsymbol{e}_h)^\top \mathrm{MLP}^{\mathrm{mod}}(\boldsymbol{e}_m) + b_{h\text{-}m},$$

$$\boldsymbol{T} = \operatorname*{arg\,max}_{\boldsymbol{T} \in \mathcal{T}(\boldsymbol{x})} \sum_{h,m} W_{h,m} T_{h,m}.$$



$$p_\theta(\boldsymbol{T}|x) = \frac{\exp(\sum_{h,m} W_{h,m} T_{h,m})}{\sum_{T' \in \mathcal{T}(x)} \exp(\sum_{h,m} W_{h,m} T'_{h,m})}.$$

- Optimization with
  - Monte-Carlo estimates

$$\log p_\theta(y^i|x^i) = \log \mathbb{E}_{\boldsymbol{T} \sim p_\theta(\boldsymbol{T}|x^i)}[p_\theta(y^i|\boldsymbol{T}, x^i)]$$
$$\geq \mathbb{E}_{\boldsymbol{T} \sim p_\theta(\boldsymbol{T}|x^i)}[\log p_\theta(y^i|\boldsymbol{T}, x^i)]. \quad (4$$

  - Gumbel perturbation

$$G_{h,m} \sim \mathcal{G}(0,1),$$
$$\widetilde{\boldsymbol{W}} = \boldsymbol{W} + \boldsymbol{G},$$
$$\boldsymbol{T} = \operatorname*{arg\,max}_{\boldsymbol{T} \in \mathcal{T}(x)} \sum_{h,m} T_{h,m} \widetilde{W}_{h,m}.$$

  - Softmax instead of argmax (in Eisner), though the output T is not valid dependency trees anymore, but a soft selection of arcs instead.

Perturb-and-Parse: Results

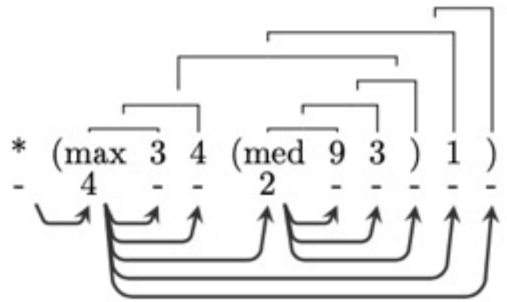


Figure 2: An example from the ListOps dataset. Numbers below operation tokens are valencies. **(top)** the original unlabelled phrase-structure. **(bottom)** our dependency conversion: each dependency represents either an operand to argument relation or a closing parenthesis relation.

	Acc.	Att.
Latent tree - $G = 0$		
Forward relaxed	98.1	83.2
Straight-Through	70.8	33.9
Latent tree - MC training		
Forward relaxed	**99.6**	**99.7**
Straight-Through	77.0	83.2

Table 1: ListOps results: tagging accuracy (Acc.) and attachment score for the latent tree grammar (Att.).

	Acc.	#Params
Yogatama et al. (2017)		
*100D SPINN	80.5	2.3M
Maillard et al. (2017)		
LSTM	81.2	161K
*Latent Tree-LSTM	81.6	231K
Kim et al. (2017)		
No Intra Attention	85.8	-
Simple Simple Att.	86.2	-
*Structured Attention	86.8	-
Choi et al. (2018)		
*100D ST Gumbel Tree	82.6	262K
*300D ST Gumbel Tree	85.6	2.9M
*600D ST Gumbel Tree	86.0	10.3M
Niculae et al. (2018)		
Left-to-right Trees	81.0	-
Flat	81.7	-
Treebank	81.7	-
*SparseMAP	81.9	-
Liu and Lapata (2018)		
175D No Attention	85.3	600K
*100D Projective Att.	86.8	1.2M
*175D Non-projective Att.	86.9	1.1M
This work		
No Intra Attention	84.4	382K
Simple Intra Att.	83.8	582K
*Latent Tree + 1 GCN	85.2	703K
*Latent Tree + 2 GCN	86.2	1M

Table 2: SNLI results and number of network parameters (discarding word embeddings). Stars indicate latent tree models.

Perturb-and-Parse: Results

(a)

Socher et al. (2013)	
Bigram Naive Bayes	83.1
Niculae et al. (2018)	
CoreNLP	83.2
*Latent tree	84.7
This work	
CoreNLP	83.8
*Latent tree	84.6

(b)

	Acc.
Williams et al. (2018a)	
300D LSTM	69.1
*300D SPINN	66.9
300D Balanced Trees	68.2
*300D ST Gumbel Tree	69.5
*300D RL-SPINN	67.3
This work	
No Intra Attention	68.1
*Latent tree + 1 GCN	71.5
*Latent tree + 2 GCN	73.0

(c)

	Match	**Mis.**
Baselines		
No Intra Att	68.5	68.9
Simple Intra Att	67.9	68.4
Left-to-right trees		
1 GCN	71.2	71.8
2 GCN	72.3	71.1
Latent head selection model		
1 GCN	69.0	69.4
2 GCN	68.7	69.6
Latent tree model		
1 GCN	71.9	71.7
2 GCN	73.2	72.9

Table 3: **(a)** SST results. Stars indicate latent tree models. **(b)** MultiNLI results. Stars indicate latent tree models. **(c)** Ablation tests on MultiNLI (results on the matched and mismatched development sets).

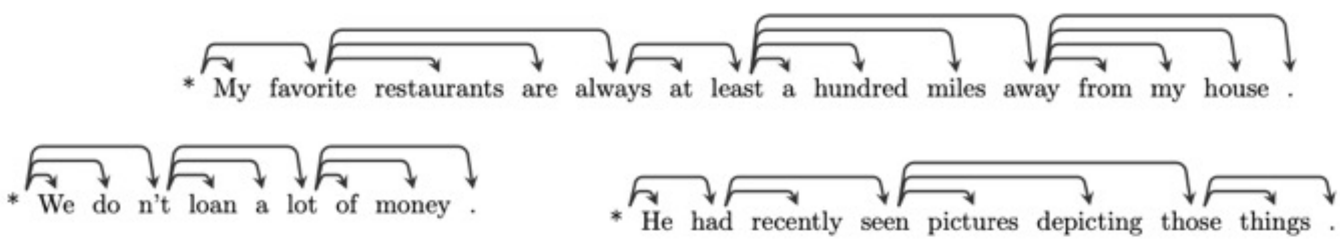


Figure 3: Examples of trees induced on the matched development set of MultiNLI, the model using 2 GCN layers.

R2D2: Recursive Transformer based on Differentiable Tree for Interpretable Hierarchical Language Modeling

ACL 2021

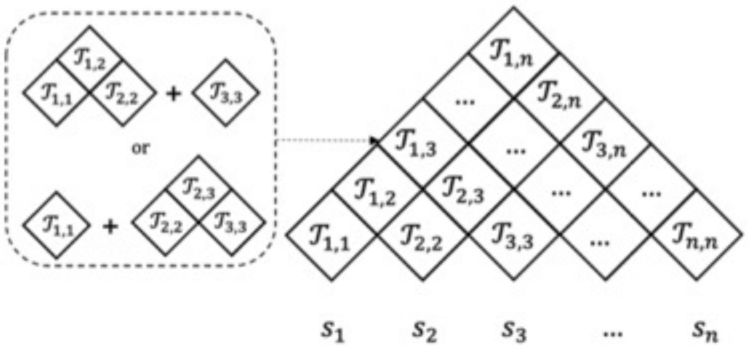
Xiang Hu[†*] **Haitao Mi**[†*] **Zujie Wen**[†] **Yafang Wang**[†]
Yi Su[†] **Jing Zheng**[†] **Gerard de Melo**[‡]
Ant Financial Services Group[†]
{aaron.hx, haitao.mi, zujie.wzj, yafang.wyf, yi.su, jing.zheng}
@alibaba-inc.com[†]
Hasso Plattner Institute / University of Potsdam[‡]
gdm@demelo.org[‡]

R2D2: Motivations

- Human language is assumed to possess a recursive hierarchical structure.
- Pretrained-LMs
  - has fixed depth and requires positional embeddings
  - do not explicitly reflect the hierarchical structures
- Fully differentiable CKY
  - is $O(N^3)$ and hard to scale up

- Contributions:
  - Recursive Transformers learn both representations and structures
  - an efficient optimization algorithm O(n) to scale up
  - an effective training objective

R2D2: Models



$$c^k_{i,j},\, p^k_{i,j} = f(e_{i,k}, e_{k+1,j})$$
$$\widetilde{p}^k_{i,j} = p^k_{i,j}\ \widetilde{p}_{i,k}\ \widetilde{p}_{k+1,j}$$
$$\boldsymbol{\alpha}_{i,j} = \textsc{Gumbel}(\log(\widetilde{\mathbf{p}}_{i,j}))$$
$$e_{i,j} = [c^i_{i,j}, c^{i+1}_{i,j}, \ldots, c^{j-1}_{i,j}]\boldsymbol{\alpha}_{i,j}$$
$$[p_{i,j}, \widetilde{p}_{i,j}] = \boldsymbol{\alpha}^{\intercal}_{i,j}[\boldsymbol{p}_{i,j}, \widetilde{\boldsymbol{p}}_{i,j}]$$

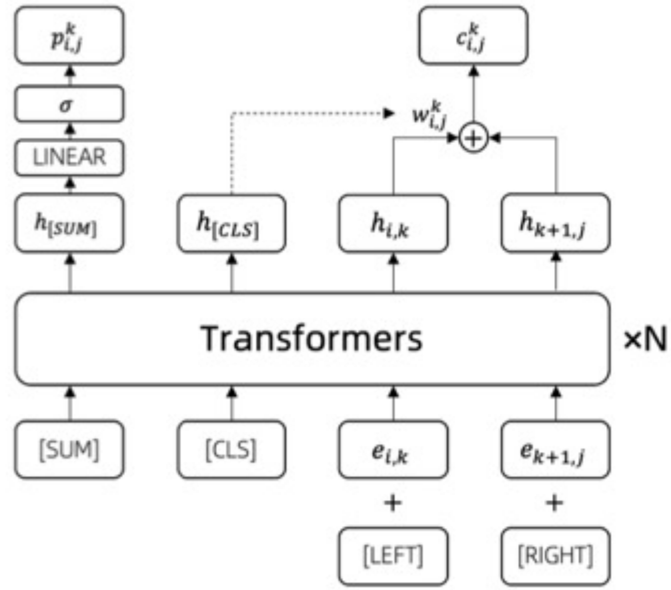


Figure 3: Recursive Transformer-based encoder.

$$\boxed{A_i^k = \sum_B \sum_C \sum_j \pi_{A\rightarrow BC} B_i^j C_{j+1}^k}$$

$$\min_\theta \sum_{i=1}^{n} -\log p_\theta(s_i \mid s_{1:i-1}, s_{i+1:n})$$

R2D2: Pruning

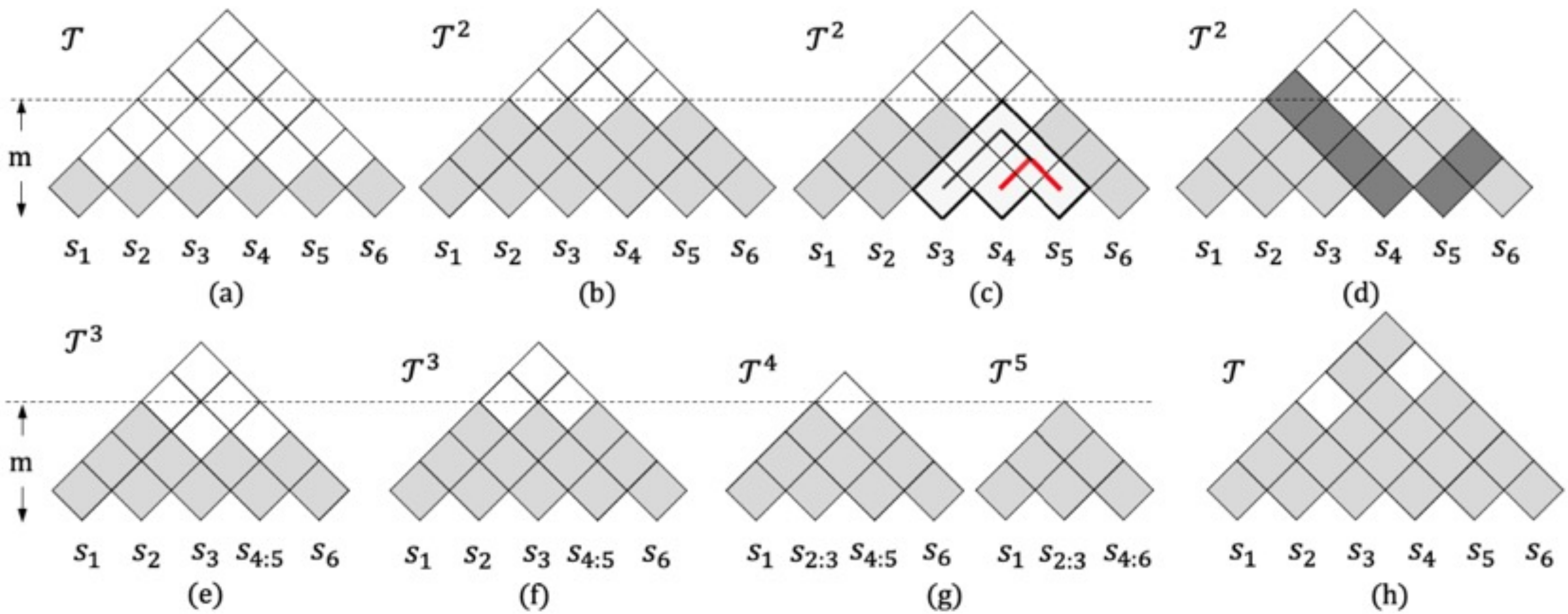


Figure 4: Example of encoding. (a) Initialized chart table. (b) Row-by-row encoding up to pruning threshold m. (c) For each cell in the m-th row, recover its subtree and collect candidate nodes, each of which must appear in the subtree and also must be in the 2nd row, e.g., the tree of $\mathcal{T}^2_{3,5}$ is within the dark line, and the candidate node is $\mathcal{T}^2_{4,5}$. (d) Find locally optimal node, which is $\mathcal{T}^2_{4,5}$ here, and treat span $s_{4:5}$ as non-splittable. Thus, the dark gray cells become prunable. (e) Construct a new chart table $\mathcal{T}^3$ treating cell $\mathcal{T}^2_{4,5}$ as a new *terminal* node and eliminating the prunable cells. (f) Compute empty cells in m-th row. (g) Keep pruning and growing the tree until no further empty cells remain. (h) Final discrete chart table.

R2D2: Experiments

	#param	#layer	#epoch	cplx	PPPL
BERT	46M	3	10	$O(n^2)$	441.42
XLNet	46M	3	10	$O(n)$	301.87
ALBERT	46M	12	10	$O(n^2)$	219.20
XLNet	116M	12	10	$O(n)$	127.74
BERT	109M	12	10	$O(n^2)$	103.54
T-LSTM (m=4)	46M	1	10	$O(n)$	820.57
Ours (m=4)	45M	3	10	$O(n)$	83.10
Ours (m=8)	45M	3	10	$O(n)$	**57.40**
BERT	46M	3	60	$O(n^2)$	112.17
XLNet	46M	3	60	$O(n)$	105.64
ALBERT	46M	12	60	$O(n^2)$	71.52
XLNet	116M	12	60	$O(n)$	59.74
BERT	109M	12	60	$O(n^2)$	**44.70**
Ours (m=4)	45M	3	60	$O(n)$	55.70
Ours (m=8)	45M	3	60	$O(n)$	54.60

Table 1: Comparison with state-of-the-art models trained from scratch on WikiText-2 with different settings (number of Transformer layers and training epochs). m is the pruning threshold.

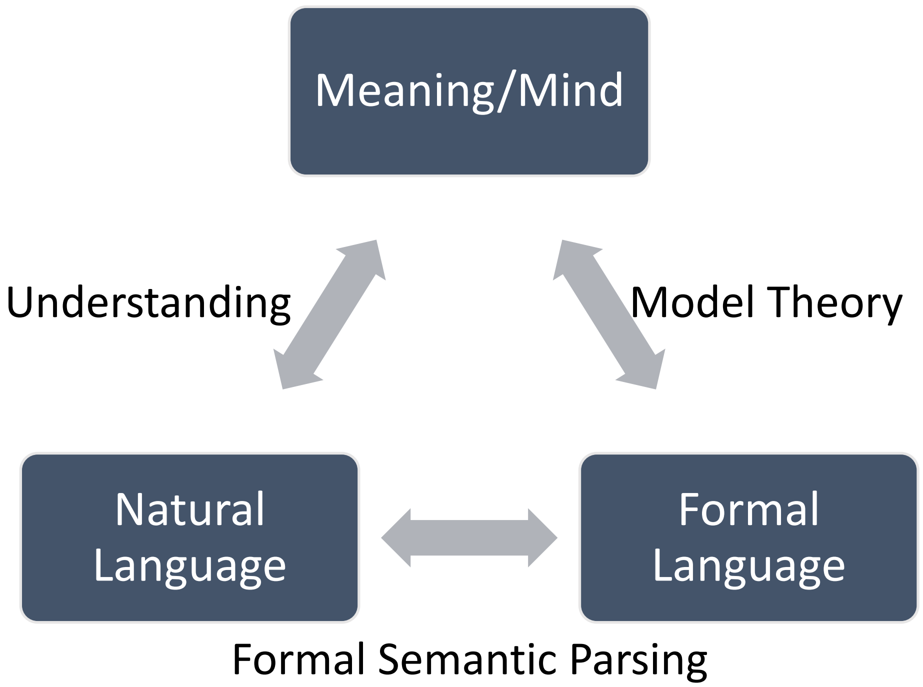
Model	cplx	WSJ F_1(M)	WSJ F_1	CTB F_1
Left Branching (W)	$O(n)$	-	8.15	11.28
Right Branching (W)	$O(n)$	-	39.62	27.53
Random Trees (W)	$O(n)$	-	17.76	20.17
BERT-MASK (WP)	$O(n^4)$	-	37.39	33.24
ON-LSTM (W)	$O(n)$	50.0†	47.72	24.73
DIORA (W)	$O(n^3)$	58.9†	51.42	-
C-PCFG (W)	$O(n^3)$	**60.1†**	**54.08**	**49.95**
Ours (WP)	$O(n)$	-	48.11	44.85
DIORA (WP)	$O(n^3)$	-	43.94	-
C-PCFG (WP)	$O(n^3)$	-	49.76	60.34
Ours (WP)	$O(n)$	-	**52.28**	**63.94**

Table 3: Unsupervised parsing results with word (W) or word-piece (WP) as input. Values with † are taken from Kim et al. (2019a). F_1(M) describes the max. score of 4 runs with different random seeds. The F_1 column shows results of our runs with a random seed. The bottom three systems take word-pieces as input, and are also measured against word-piece level golden trees.

Agenda

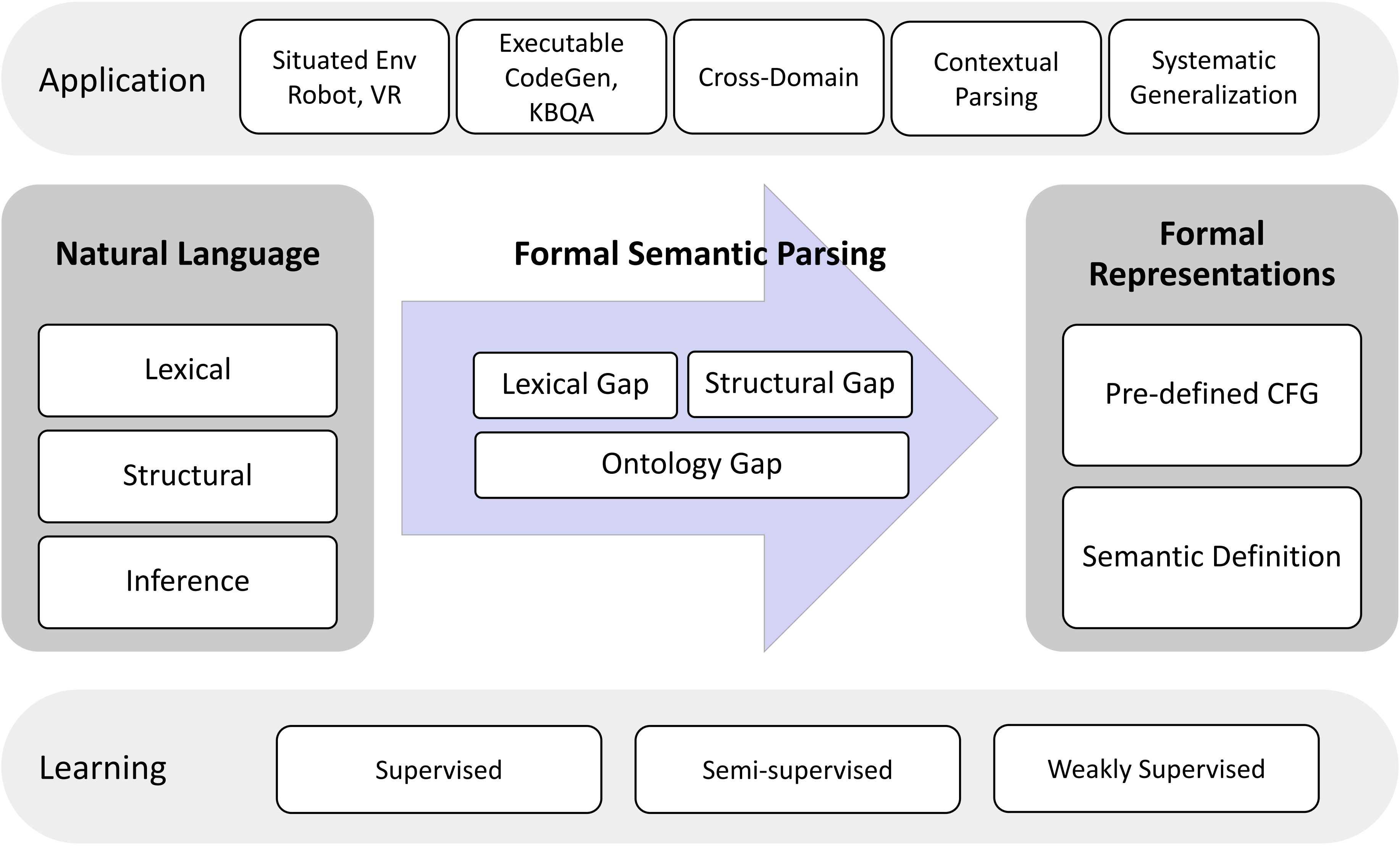
- Tutorial: Chart Parsing
- Unsupervised Grammar Induction
- Migration to Semantic Parsing
- What's Next

Semantic Parsing as A Meaning Surrogate



- Formal Language is artificial
  - targeted and specialized
  - not exactly equivalent to NL
  - Analysis of NL by FL is barking up the wrong tree
- Semantic Parsing knows little semantic
  - semantic doesn't have to be composed
  - at least, semantic without pragmatics is not meaning
- Assuming
  - semantic parsing is enough
  - within application domains, ad-hoc process requirements

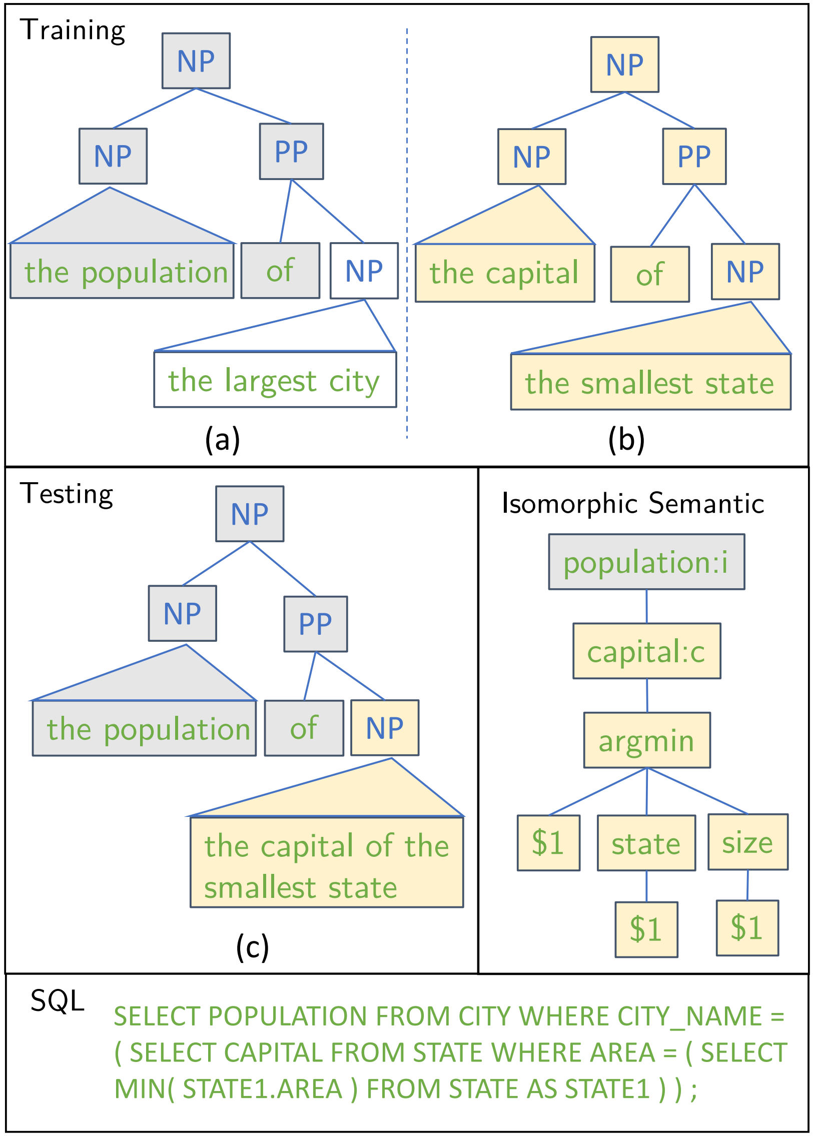
Intrinsic Features of Formal Semantic Parsing



Mapping-centric Perspective

- Characterizing Mapping Objects (1970s, 1993-2014,2016)
  - Rules or Lexicons (words to semantics, syntactic trees to semantics)
  - CCG / SCFG / HRG / AM Algebra
- Mapping as the probabilistic model: (2010-)
  - Log-linear models/hybrid trees/generative models
  - Agenda-based Parsing/Float Parser/Transition-based Parsers
  - Neural Nets
- Mapping with pattern templates(by intermediate repr.): (2014-)
  - Paraphrasing
  - Factored CCG/Sketches/Intermediate Grammar
- Alignment is found useful again (2020-)

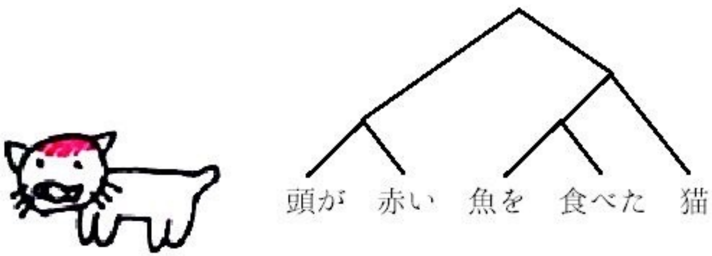
Compositional Generalization

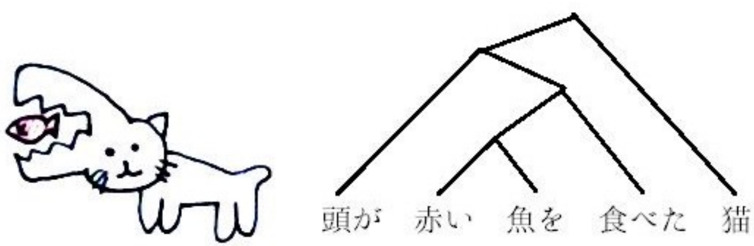


Structure Model Insufficiency

Model	ADV (CG / IID / GAP)	ATIS (CG / IID / GAP)
Baselines from (Oren et al., 2020)		
Seq-A	0.1 / 90.0 / 99.9	12.3 / 70.5 / 82.6
Bert-A	0.1 / 91.5 / 99.9	17.0 / 72.2 / 76.5
Seq-G	3.0 / 88.5 / 96.6	18.1 / 65.8 / 72.5
Bert-G	2.3 / 90.7 / 97.5	7.1 / 62.4 / 88.6
Absent Source Structure		
Seq-A	$5.9_{\pm2.6}$ / $86.2_{\pm1.0}$ / 93.2	$15.1_{\pm2.3}$ / $61.8_{\pm1.7}$ / 75.6
Seq-L	$5.2_{\pm1.6}$ / $86.2_{\pm1.9}$ / 93.9	$15.3_{\pm3.1}$ / $60.9_{\pm0.4}$ / 74.9
Seq-G	$7.8_{\pm3.7}$ / $82.3_{\pm6.2}$ / 90.5	$16.3_{\pm5.7}$ / $57.5_{\pm1.4}$ / 71.7
Bert-A*	$9.1_{\pm5.2}$ / $89.9_{\pm0.8}$ / 89.9	$29.8_{\pm0.6}$ / $67.0_{\pm1.6}$ / 55.6
Bert-L	$9.8_{\pm4.0}$ / $88.7_{\pm0.9}$ / 88.9	$19.3_{\pm0.5}$ / $62.0_{\pm0.8}$ / 68.9
Bert-G*	$7.6_{\pm3.5}$ / $87.1_{\pm1.4}$ / 91.3	$31.2_{\pm2.3}$ / $65.3_{\pm1.0}$ / 52.2
Elec-A*	$4.7_{\pm2.3}$ / $90.1_{\pm0.9}$ / 94.8	$29.0_{\pm0.8}$ / $66.7_{\pm1.1}$ / 56.6
Elec-L	$7.0_{\pm4.5}$ / $87.7_{\pm1.3}$ / 92.0	$18.6_{\pm6.4}$ / $58.2_{\pm2.8}$ / 68.1
Elec-G*	$6.0_{\pm2.2}$ / $86.8_{\pm1.0}$ / 93.1	$30.9_{\pm1.9}$ / $66.4_{\pm1.4}$ / 53.4
Latent Source Structure		
ON-A	$6.8_{\pm0.4}$ / $82.1_{\pm0.7}$ / 91.7	$22.8_{\pm3.6}$ / $63.6_{\pm1.0}$ / 64.2
ON-L	$5.0_{\pm1.5}$ / $82.3_{\pm0.0}$ / 93.9	$24.7_{\pm4.2}$ / $61.9_{\pm0.8}$ / 60.1
ON-G	$6.3_{\pm4.4}$ / $80.5_{\pm1.8}$ / 92.2	$22.2_{\pm1.6}$ / $58.9_{\pm1.6}$ / 62.4
diora-A	$3.9_{\pm0.7}$ / $66.3_{\pm7.2}$ / 94.0	$18.5_{\pm6.5}$ / $52.0_{\pm2.0}$ / 64.4
diora-L	$5.1_{\pm1.1}$ / $68.2_{\pm3.6}$ / 92.6	$17.9_{\pm2.0}$ / $50.4_{\pm1.1}$ / 64.5
diora-G	$3.3_{\pm5.1}$ / $61.5_{\pm3.0}$ / 94.6	$15.4_{\pm7.3}$ / $50.9_{\pm2.1}$ / 69.8
pcfg-A*	$2.8_{\pm1.9}$ / $81.7_{\pm0.4}$ / 96.5	$11.0_{\pm9.4}$ / $58.0_{\pm4.1}$ / 81.1
pcfg-L	-	-
pcfg-G*	$1.7_{\pm0.9}$ / $80.1_{\pm2.6}$ / 97.9	$11.8_{\pm2.8}$ / $58.3_{\pm0.4}$ / 79.8
TD-A*	$0.7_{\pm1.3}$ / $77.0_{\pm4.5}$ / 99.1	$1.4_{\pm1.4}$ / $55.0_{\pm5.0}$ / 97.4
TD-L	-	-
TD-G*	$1.5_{\pm2.0}$ / $59.6_{\pm24.2}$ / 97.5	$3.4_{\pm1.7}$ / $53.5_{\pm7.9}$ / 93.6
PnP-A	$6.3_{\pm0.2}$ / $83.6_{\pm1.0}$ / 92.5	$12.3_{\pm1.2}$ / $56.5_{\pm1.9}$ / 78.2
PnP-L	$6.2_{\pm2.4}$ / $84.3_{\pm1.4}$ / 92.6	$17.1_{\pm3.0}$ / $53.5_{\pm0.4}$ / 68.1
PnP-G	$3.3_{\pm0.5}$ / $81.9_{\pm1.4}$ / 96.0	$16.4_{\pm4.8}$ / $53.2_{\pm1.3}$ / 69.2
Given Source Structure from Berkley Parser		
Tree-A	$11.4_{\pm1.7}$ / $75.2_{\pm2.3}$ / 84.9	$16.4_{\pm2.4}$ / $57.7_{\pm0.9}$ / 71.7
Tree-L	$9.3_{\pm0.9}$ / $75.0_{\pm0.8}$ / 87.6	$16.0_{\pm1.8}$ / $52.7_{\pm3.0}$ / 69.6
Tree-G	$7.8_{\pm1.8}$ / $72.2_{\pm4.0}$ / 89.2	$17.4_{\pm4.6}$ / $53.5_{\pm1.5}$ / 67.4

Syntactic Ambiguity Meme





Mapping-centric Models

- Supervised softmax (NAACL 2021)
- Algebraic Recombination (ACL 2021)
- SpanBasedSP (ACL 2021)
- LAGr: Label Aligned Graphs (ACL 2022)

Compositional Generalization for Neural Semantic Parsing via Span-level Supervised Attention

NAACL 2021

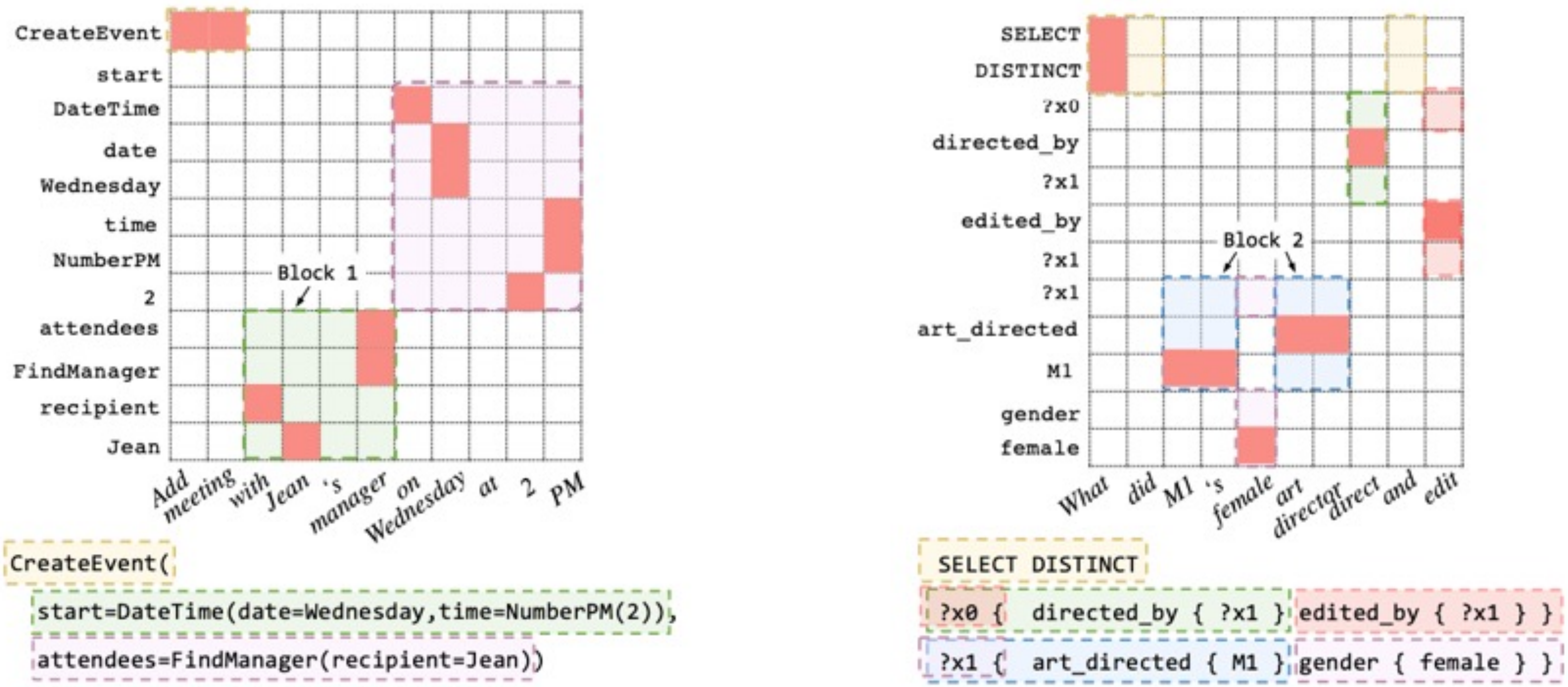
Pengcheng Yin[♠,*], **Hao Fang**[♣], **Graham Neubig**[♠], **Adam Pauls**[♣],
Emmanouil Antonios Platanios[♣], **Yu Su**[♣], **Sam Thomson**[♣], **Jacob Andreas**[♣]

[♠]Carnegie Mellon University [♣]Microsoft Semantic Machines

{pcyin,gneubig}@cs.cmu.edu

{hao.fang,adam.pauls,anthony.platanios,
yusu2,samuel.thomson,jacob.andreas}@microsoft.com

Supervised Attention: Method



(a) Utterance-LISP Expression Alignments

(b) Utterance-SPARQL Alignments

Supervised Attention: Results

$\lvert\mathbb{C}_{\text{train}}\rvert$	16		32	
Domain	$\mathbb{S}$	$\mathbb{C}$	$\mathbb{S}$	$\mathbb{C}$
BERT2SEQ	82.8 ±1.0	33.6 ±7.2	82.8 ±0.6	53.5 ±10.3
+TS (Token-level Sup.)	83.4 ±0.7	39.7 ±1.3	83.2 ±0.3	59.9 ±1.6
+SS (Span-level Sup.)	83.9 ±0.2	**46.8** ±1.2	83.5 ±0.7	**61.7** ±2.2
COARSE2FINE (DL18)	83.0 ±1.0	40.6 ±7.0	83.6 ±0.6	54.6 ±6.8
+TS (Token-level Sup.)	83.7 ±0.5	44.6 ±1.5	83.1 ±1.0	60.7 ±2.5
+SS (Span-level Sup.)	83.8 ±0.4	**47.4** ±2.1	83.7 ±1.0	**61.9** ±1.8

Table 1: TEST. accuracies on the SMCALFLOW-CS Compositional Skills

Model	*Query Split*		*i.i.d. Split*	
	DEV.	TEST.	DEV.	TEST.
Oren et al. (2020)	28.9	34.4	78.4	74.5
+ Token-level Sup.	31.2 ±1.2	34.5 ±0.9	76.7 ±0.6	72.5 ±1.6
+ Span-level Sup.	31.1 ±0.6	35.0 ±2.0	78.4 ±0.8	74.0 ±0.5

Table 3: Accuracies and standard deviation on the ATIS text-to-SQL query (program template) and standard i.i.d. split splits. Results averaged over five random runs.

Split	MCD_1			MCD_2			MCD_3			Average
	C	R	All	C	R	All	C	R	All	
T5-BASE	**55.8** ±4.8	77.4 ±4.7	**62.4** ±4.5	34.8 ±2.9	29.4 ±2.5	33.0 ±2.4	**21.6** ±8.6	34.4 ±2.8	23.0 ±1.7	39.5
+ TS	44.9 ±4.7	**86.4** ±2.4	57.7 ±3.4	32.4 ±3.1	32.7 ±1.4	32.5 ±2.1	14.3 ±1.5	36.6 ±1.7	22.0 ±0.7	37.4
+ SS	48.2 ±4.4	80.5 ±2.2	58.2 ±2.8	34.8 ±2.3	**36.4** ±2.8	**35.4** ±1.6	14.6 ±2.1	**40.1** ±3.5	23.8 ±1.0	39.1

Table 2: Mean Test Accuracies on CFQ MCD splits with 95% confidence interval, for **C**onjunctive, **R**ecursive, and **All** the samples. The last column lists averaged accuracies for the three splits. **Bold** results have p-values ≤ 0.01 when comparing to other systems in the same category.

Span-based Semantic Parsing for Compositional Generalization
ACL 2021

Jonathan Herzig[1] **Jonathan Berant**[1,2]

[1]Blavatnik School of Computer Science, Tel-Aviv University
[2]Allen Institute for Artificial Intelligence
{jonathan.herzig,joberant}@cs.tau.ac.il

SpanBasedSP: Methods

- Spans mapped to: domain categories, join, and Ø
  - Hard-EM for training without tree supervision
  - CKY-style inference

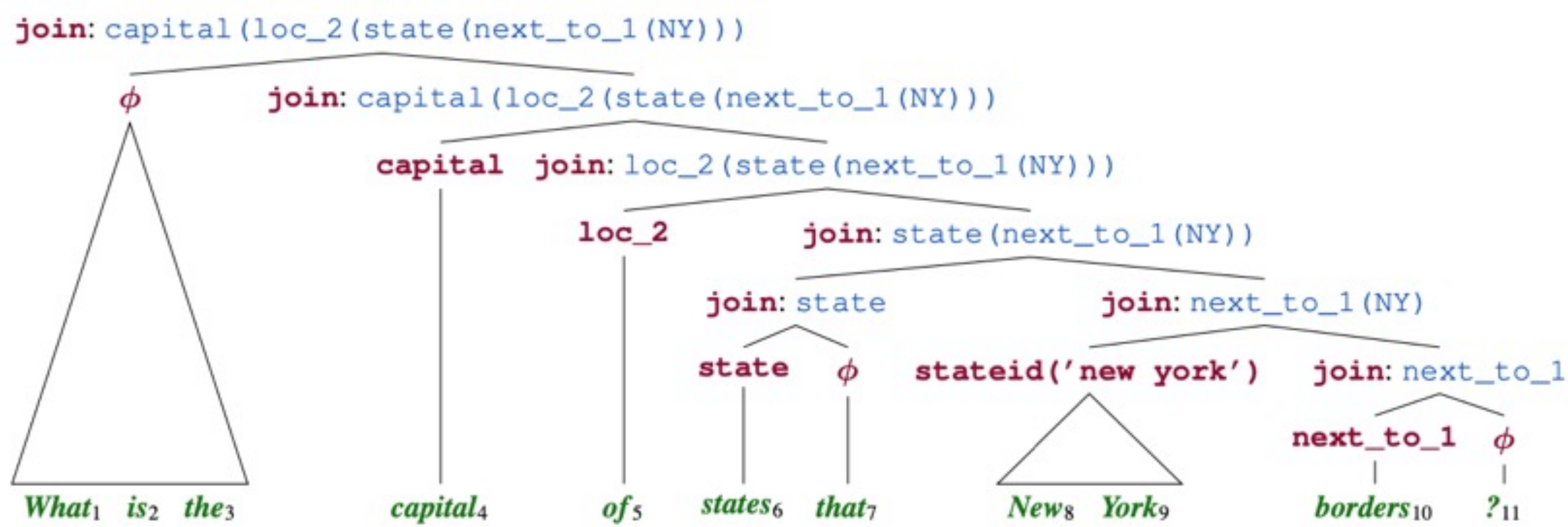


Figure 1: An example span tree. Nodes are annotated with categories (in bold). A node with a category `join` over the span (i, j), is annotated with its sub-program $z_{i:j}$. We abbreviate `stateid('new york')` to `NY`.

SpanBasedSP: Results

Model	SCAN-SP						CLEVR				GeoQuery					
	IID		RIGHT		AROUNDRIGHT		IID		CLOSURE		IID		TEMPLATE		LENGTH	
	dev	**test**	**dev**	**test**	**dev**	**test**	**dev**	**test**	**dev**	**test**	**dev**	**test**	**dev**	**test**	**dev**	**test**
Seq2Seq	100	99.9	100	11.6	100	0.0	100	100	100	59.5	83.3	78.5	71.6	46.0	86.7	24.3
+ELMo	100	100	100	54.9	100	41.6	100	100	100	64.2	83.3	79.3	83.3	50.0	86.7	25.7
BERT2Seq	99.9	100	99.9	77.7	99.9	95.3	100	100	100	56.4	88.3	81.1	85.0	49.6	90.0	26.1
GRAMMAR	100	100	100	0.0	100	4.2	100	100	100	51.3	78.3	72.1	76.7	54.0	81.7	24.6
BART	100	100	100	50.5	100	100	100	100	100	51.5	93.3	87.1	86.7	67.0	90.0	19.3
End2End	-	-	-	-	-	-	99.9	99.8	99.9	63.3	-	-	-	-	-	-
SpanBasedSP	100	100	100	**100**	100	**100**	97.0	96.7	98.9	**98.8**	88.3	86.1	93.3	**82.2**	95.0	**63.6**
-lexicon	100	100	100	**100**	100	**100**	99.4	99.3	98.5	88.6	88.3	78.9	86.7	65.9	90.0	41.4
-non projective	-	-	-	-	-	-	-	-	-	-	85.0	80.0	90.0	80.2	93.3	59.3
+gold trees	100	100	100	100	100	100	100	96.8	100	96.7	91.2	86.4	100	81.8	96.7	68.6

Table 2: Denotation accuracies for all models, including SpanBasedSP ablations. For both CLEVR splits, SpanBasedSP only trains on $10K$ examples, in comparison to $695K$ for the baselines.

Learning Algebraic Recombination for Compositional Generalization

ACL 2021

Chenyao Liu[1*] **Shengnan An**[2*] **Zeqi Lin**[3†] **Qian Liu**[4*] **Bei Chen**[3]
Jian-Guang LOU[3] **Lijie Wen**[1†] **Nanning Zheng**[2] **Dongmei Zhang**[3]
[1] School of Software, Tsinghua University [2] Xi'an Jiaotong University
[3] Microsoft Research Asia [4] Beihang University
{liucy19@mails, wenlj@}.tsinghua.edu.cn
{an1006634493@stu, nnzheng@mail}.xjtu.edu.cn
{Zeqi.Lin, beichen, jlou, dongmeiz}@microsoft.com
qian.liu@buaa.edu.cn

AlgeRecom: Motivations

- Compositional generalization requires algebraic recombination
  - model semantic parsing as a homomorphism between algebra
- Syntactic algebra
  - $L =< L, (f_\gamma)_\gamma \in \Gamma >, f_\gamma : L^k \to L$
  - latent and learnt from data
- Semantic algebra
  - M=<M,G>
  - by enumerating all available semantic primitives and operations
- Homomorphism mapping

$$m(f_\gamma(e_1, ..., e_k)) = g_\gamma(m(e_1), ..., m(e_k)).$$

AlgeRecom: Methods

- Model: composer+interpreter
- Composer: Latent Tree-LSTM
- Interpreter:
  - lexical nodes
  - algebraic nodes

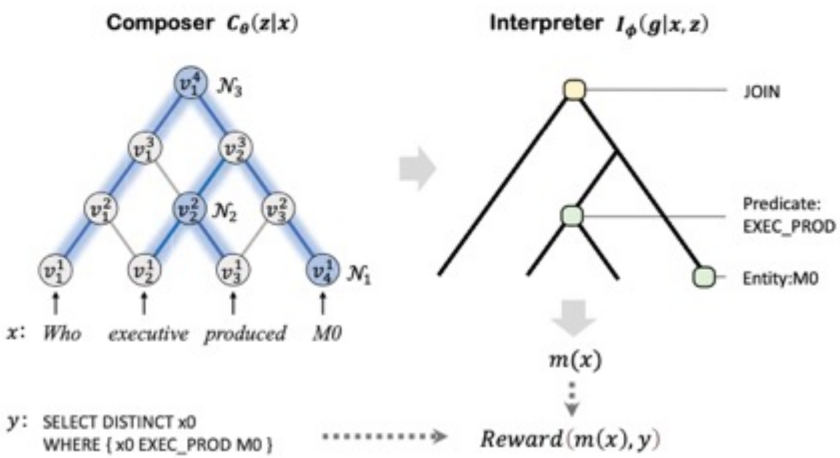


Figure 2: An overview of LEAR: (1) Composer $C_\theta(z|x)$ is a neural network based on latent Tree-LSTM, which produces the latent syntax tree z of input expression x; (2) Interpreter $I_\phi(g|x,z)$ is a neural network that assigns a semantic operation for each non-terminal node in z.

Operation	Args$[t_1, t_2]\rightarrow$ Result Type	Example
$\wedge(t_1, t_2)$	[P, P]→P	Who [**direct** and **act**] M0?
	[E, E]→E	Who direct [**M0** and **M1**]?
	[A, A]→A	Is M0 an [**Italian** **female**]?
	[A, E]→E [E, A]→E	Is [**M0** an **Italian female**]?
	[A, P]→P [P, A]→P	Is M0 M3's [**Italian** **editor**]?
JOIN(t_1, t_2)	[E, P]→E [P, E]→E	Is M0 an [**editor** of **M1**]?
	[A, P]→E [P, A]→E	Who [**marries** an **Italian**]?

Table 1: Semantic operations in CFQ. A/P/E represents Attribute/Predicate/Entity.

AlgeRecom: Setup

- Training:
  - REINFORCE with
  - Logic-based reward

$$R_1(\tau) = \text{Jaccard-Sim}(S_{m(x)}, S_y)$$

  - Primitive-based reward

$$R_2(\tau) = \text{Jaccard-Sim}(S'_{m(x)}, S'_y)$$

- Training techniques
  - space pruning
  - curriculum learning

CFQ

```
x  "Did a male film director edit and direct M0?"
y  SELECT count ( * ) WHERE {
       ?x0 ns:film.director.film M0 .
       ?x0 ns:film.editor.film M0 .
       ?x0 ns:people.person.gender m_05zppz }
```

COGS

```
x  "Charlotte was given the cake on a table."
y  cake(x_4) ; give.recipient (x_2, Charlotte)
   AND give.theme(x_2,x_4)
   AND cake.nmod.on(x_4, x_7)
   AND table(x_7)
```

Figure 3: Examples of CFQ and COGS.

Statistics	CFQ	COGS
Train Size	95,743	24,155
Dev Size	11,968	3,000
Test Size	11,968	21,000
Vocab Size	96	740
Avg Input Len (Train/Test)	13.5/15.1	7.5/9.8
Avg Output Len (Train/Test)	27.7/34.0	43.6/67.6
Input Pattern Coverage[a]	0.022	0.783
Output Pattern Coverage	0.045	0.782

Table 2: Dataset statistics.

AlgeRecom: Results

Models	MCD-MEAN
LSTM+Attention (Keysers et al., 2019)	14.9±1.1
Transformer (Keysers et al., 2019)	17.9±0.9
Universal Transformer (Keysers et al., 2019)	18.9±1.4
Evolved Transformer (Furrer et al., 2020)	20.8±0.7
T5-11B (Furrer et al., 2020)	40.9±4.3
T5-11B-mod (Furrer et al., 2020)	42.1±9.1
Neural Shuffle Exchange (Furrer et al., 2020)	2.8±0.3
CGPS (Furrer et al., 2020; Li et al., 2019)	7.1±1.8
HPD (Guo et al., 2020b)	67.3±4.1
LEAR	**90.9±1.2**
w/o Abstraction	85.4±4.5
w/o Semantic locality	87.9±2.7
w/o Primitive-based reward	85.3±7.8
w/o Curriculum learning	71.9±15.4
w/o Tree-LSTM	30.4±3.2

Table 3: Accuracy on three splits (MCD1/MCD2/M

Model	Acc
Transformer (Kim and Linzen, 2020)	35 ± 6
LSTM (Bi) (Kim and Linzen, 2020)	16 ± 8
LSTM (Uni) (Kim and Linzen, 2020)	32 ± 6
LEAR	**97.7 ± 0.7**
w/o Abstraction	94.5 ± 2.8
w/o Semantic locality	94.0 ± 3.6
w/o Tree-LSTM	80.7 ± 4.3

Table 4: Accuracy on COGS benchmark.

Model	Acc
Seq2Seq (Herzig and Berant, 2020)	46.0
BERT2Seq (Herzig and Berant, 2020)	49.6
GRAMMAR (Herzig and Berant, 2020)	54.0
PDE (Guo et al., 2020c)	81.2
SpanBasedSP (Herzig and Berant, 2020)	82.2
LEAR	**84.1**

Table 5: Accuracy on GEO benchmark.

LAGr: Label Aligned Graphs for Better Systematic Generalization in Semantic Parsing

ACL 2022

Dóra Jámbor*
Mila - Quebec AI Institute
McGill University
dora.jambor@mail.mcgill.ca

Dzmitry Bahdanau
ServiceNow Research
Mila - Quebec AI Institute
McGill University
Canada CIFAR AI Chair

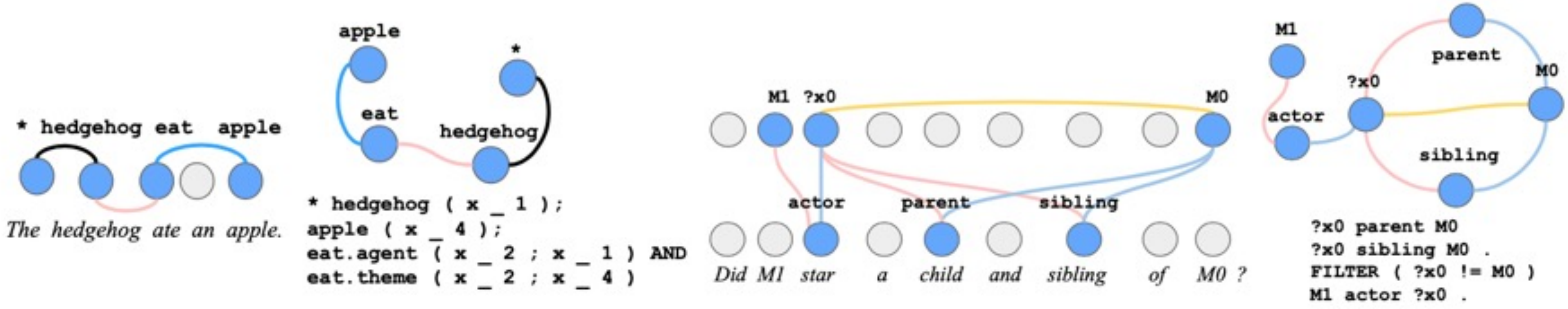
LAGr: Motivations

- Intuition: A model
  - that predicts such aspects of meaning independently
  - can be better at learning context-insensitive rules.
- Semantic parses as graphs haven't been tested on systematic generalization
- Existing methods (e.g. SpanBasedSP) raise complexities and rigidities against seq2seq models.

- Latent Alignment
  - inferred with an MAP algorithm involving minimum cost bipartite matching problems with the Hungarian algorithm

LAGr: Formalization

- Sentence: $x = x_1, x_2, \dots, x_N$
- The graph Γ_a
  - with M = L * N nodes arranged in L layers
  - $\Gamma_a = (z, \xi), z \in V_n^M, \xi \in V_e^{M \times M}$ indicating node labels and edge labels



(a) COGS (b) CFQ

Figure 2: Aligned and unaligned graphs for COGS (a) and CFQ (b). For COGS, pink, blue and black denote agent, theme and **article** edges, respectively. For CFQ, yellow, pink and blue mark FILTER, agent, theme edges. Grey nodes mark null nodes, and ⋆ denotes the definite article. The aligned graph for CFQ is provided for illustration purposes, and was not used for training. See Section 4 for the learned aligned graphs.

LAGr: Weakly-supervised Formalization

- For weakly-supervised cases, Γ_{na} = (s, e) is assumedly produced by permuting the columns of a latent aligned graph Γ_a.
- The permutation denoted as a
  - a_j is the index of column in Γ_a that becomes the j-th column of Γ_{na}
- Then the (intractable) model is

$$p(e, s|x) = \sum_a \sum_z \sum_\xi p(e, s, a, z, \xi|x)$$
$$= \sum_a p(a) \prod_j p(z_{a_j} = s_j|x)$$
$$\prod_j \prod_k p(\xi_{a_j a_k} = e_{jk}|x),$$

LAGr: MAP inference

- Use the MAP alignment: $\hat{a} = \arg\max_a p(a \mid e, s, x)$
- Training Objective:

$$p(e, s|\hat{a}, x) = \prod_j p(z_{\hat{a}_j} = s_j|x) \prod_j \prod_k p(\xi_{\hat{a}_j,\hat{a}_k} = e_{jk}|x).$$

- MAP detail:

$$\begin{aligned}
\hat{a} &= \arg\max_a p(a|e, s, x) \\
&= \arg\max_a \log p(s|a, x) + \log p(e|a, x) \\
&= \arg\max_a \Big[\sum_j \log p(z_{a_j} = s_j|x) \\
&\quad + \sum_j \sum_k \log p(\xi_{a_j,a_k} = e_{j,k}|x)\Big]
\end{aligned}$$

LAGr: Results (1/2)

	Exact match accuracy (%)		
	train	test	gen
LSTM+Attn ♢	-	99.	16. (±8.)
Transformer ♢	-	96.	35. (±6.)
LSTM+Attn ♡	-	-	51. (±5.)
Transformer ♣	-	-	**81.** (±1.)
LSTM + Lex: Simple ♡	-	-	**82.** (±1.)
LSTM + Lex: PMI ♡	-	-	**82.** (±0.)
LSTM + Lex: IBMM2 ♡	-	-	**82.** (±0.)
LSTM+Attn (ours)	100 (±0.0)	99.6 (±0.2)	26.1 (±6.8)
LSTM_{sh} strongly-supervised LAGr	100 (±0.0)	99.9 (±0.1)	39.0 (±9.1)
LSTM_{sep} strongly-supervised LAGr	100 (±0.0)	100 (±0.0)	71.4 (±2.9)
Transformer (ours)	100 (±0.0)	99.8 (±0.0)	80.6 (±1.4)
Transformer_{sh} strongly-supervised LAGr	100 (±0.0)	100 (±0.0)	80.2 (±1.4)
Transformer_{sep} strongly-supervised LAGr	100 (±0.0)	99.9 (±0.1)	**82.5** (±2.9)
Transformer_{sep} weakly-supervised LAGr	100 (±0.0)	99.9 (±0.0)	80.7 (±2.5)
Transformer_{sep} weakly-supervised LAGr + Retrain	100 (±0.0)	99.9 (±0.0)	**82.3** (±2.3)

Table 1: Average exact match accuracy and standard deviation on COGS. **Bottom**: reproduced seq2seq baselines and LAGr. **Middle:** Seq2seq baselines including the original results by Kim and Linzen (2020a) ♢, best Transformer baseline by Csordás et al. (2021) ♣, and the best LSTM baseline by Akyürek and Andreas (2021) ♡. We also show results by the lexicon-based approach by Akyürek and Andreas (2021).

LAGr: Results (2/2)

	Graph Accuracy					
	Random		Mean MCD	MCD1	MCD2	MCD3
	train	test	test	test	test	test
HPD ♠	-	-	67.3 (∓4.1)	72.0 (∓7.5)	66.1 (∓6.4)	63.9 (∓5.7)
HPD w/o Hierarchical Mechanism ♠	-	-	-	21.3	6.4	10.1
T5-small + IR ♢	-	-	47.9	-	-	-
LSTM + Attn ♡	-	97.4 (∓0.3)	14.9 (∓1.1)	28.9 (∓1.8)	5.0 (∓0.8)	10.8 (∓0.6)
Transformer ♡	-	98.5 (∓0.2)	17.9 (∓0.9)	34.9 (∓1.1)	8.2 (∓0.3)	10.6 (∓1.1)
Universal Transformer ♡	-	98.0 (∓0.3)	18.9 (∓1.4)	37.4 (∓2.2)	8.1 (∓1.6)	11.3 (∓0.3)
Evol. Transformer ♣	-	-	20.8 (∓0.7)	42.4 (∓1.0)	9.3 (∓0.8)	10.8 (∓0.2)
LSTM + Simplified SPARQL ♠	-	-	26.1	42.2	14.5	21.5
Transformer + Simplified SPARQL ♠	-	-	31.4	53.0	19.5	21.6
T5-small from scratch ♢	-	-	20.8	-	-	-
T5-small from scratch + IR ♢	-	-	22.6	-	-	-
Transformer$_{sh}$ weakly sup. LAGr, $K = 1$	100 (∓0.0)	99.5 (∓0.2)	**38.2** (∓2.7)	**65.2** (∓2.6)	26.4 (∓3.2)	23.0 (∓2.0)
Transformer$_{sh}$ weakly sup. LAGr, $K = 5, \sigma = 10$	100 (∓0.0)	99.7 (∓0.0)	**39.5** (∓3.2)	62.8 (∓4.0)	**30.3** (∓2.7)	**25.4** (∓2.7)

Table 2: Average graph accuracy and standard deviation of weakly-supervised LAGr on CFQ (**bottom**). **Middle:** results by several seq2seq baselines from prior work (Keysers et al. (2020) ♡, Furrer et al. (2020) ♣). **Top:** results not directly comparable to LAGr: Hierarchical Poset Decoding (Guo et al., 2020) ♠, and pretrained T5-small seq2seq model with intermediate representations (IR) (Herzig et al., 2021) ♢. Approaches other than LAGr report the average exact match accuracy with 95% confidence intervals.

Agenda

- Tutorial: Chart Parsing
- Unsupervised Grammar Induction
- Migration to Semantic Parsing
- What's Next

What's Next

- Non-Isomorphic Semantic Representations Analysis
- Mapping with Latent Alignments
- Mapping through NPDA model
- Domain Adaptation following Game-theoretic Semantics and Game-based WSD

Hungarian Latent Alignments on Hungarian

Table: Province of Alessandria

City (c1)	*Population (c2)*	*Area (km²) (c3)*	...
Alessandria	94191	203.97	...
Casale Monferrato	36039	86.32	...
Novi Ligure	28581	54.22	...
Tortona	27476	99.29	...
Acqui Terme	20426	33.42	...

Question: ① How many ② cities have ③ at least ④ 25,000 ⑤ people?

Target Logical Form:

```
SELECT count(c1) FROM w WHERE c2_number>=25000
```

Answer: 4

Table: Bulgaria at the 1988 Winter Olympics

Athlete (c1)	*Total Time (c2)*	*Total Rank (c3)*	...
Stefan Shalamanov	1:52.37	23	...
Borislav Dimitrachkov	1:50.81	19	...
Petar Popangelov	1:46.34	16	...

Question: ① Who has ② the highest ③ rank ?

Target Logical Form:

```
SELECT c1 FROM w ORDER BY c3_number LIMIT 1
```

Answer: Petar Popangelov

weight-policy	train	EM	Pred
squall-0			
sparsemax	42.6%	35.6%	36.6%
hungarian-sup	86.4%	39.6%	40.6%
hungarian-reg	89.0%	41.1%	42.3%
oracle-sup	86.6%	41.5%	**42.7%**

squall-0			
softmax	67.9%	39.5%	40.6%
Hungarian-sup	85.7%	38.7%	40.0%
hungarian-reg	86.7%	42.1%	**43.5%**
oracle-sup	89.5%	42.1%	43.4%

Thanks!