A Parsing Model and a Generation Model for Graph-Structured Syntacto-Semantic Representations

Weiwei Sun

Institute of Computer Science and Technology
Peking University

June 18, 2018

Overview

SHRG-based Parsing

Generation via DAG Transduction

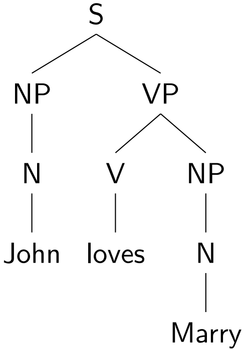
Outline

SHRG-based Parsing

Generation via DAG Transduction

Context-free Grammar (CFG) for Strings

John loves Marry.



Context-free Grammar (CFG) for Strings

John loves Marry.

- S
- NP + VP
- N + VP
- John + VP
- John + V + NP
- John + loves + NP
- John + loves + N
- John + loves + Marry

Hyperedge Replacement Grammar

- A context-free rewriting system for graphs
- Hyperedge: an extension of a normal edge; a hyperedge can connect to more than two nodes or only one node.
- The rewriting process replaces a nonterminal hyperedge with a graph fragment

Hyperedge Replacement Grammar

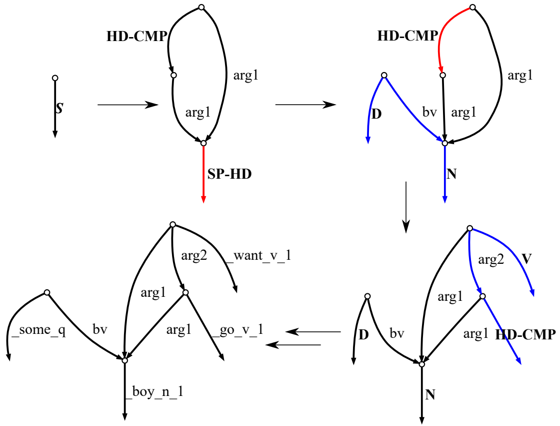
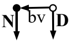


Figure 1: “Some boys want to go.”

Synchronous Hyperedge Replacement Grammar

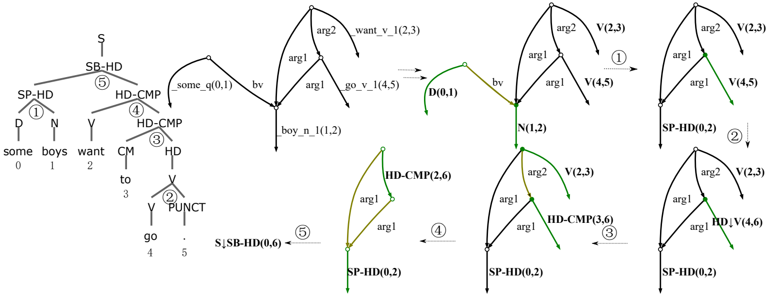
- Mapping between CFG, a string grammar, and HRG, a graph grammar
- A mathematically sound framework to construct semantic graphs: when a coherent CFG derivation is ready, we can *interpret* it using the corresponding HRG and get a semantic graph.

	①	②	③	④	⑤
Shared LHS	SP-HD	HD↓V	HD-CMP	HD-CMP	S↓SP-HD
RHS (syntax)	D + N	V + PUNCT	CM + HD↓V	V + HD-CMP	SP-HD + HD-CMP
RHS (semantics)					

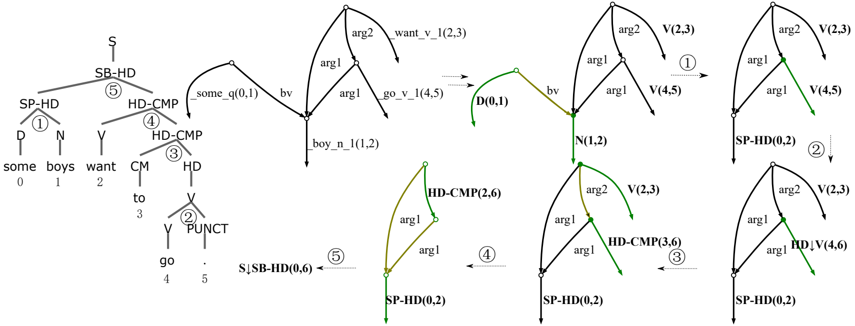
Grammar Extraction

We propose an algorithm to derive SHRGs from *graph-annotated* sentences.

- It requires and only requires alignments between edges and surface strings
- *Trees* are also required, but don't have to be *gold* syntactic ones.
- It recursively identifies a subgraph and replace it into an hyperedge

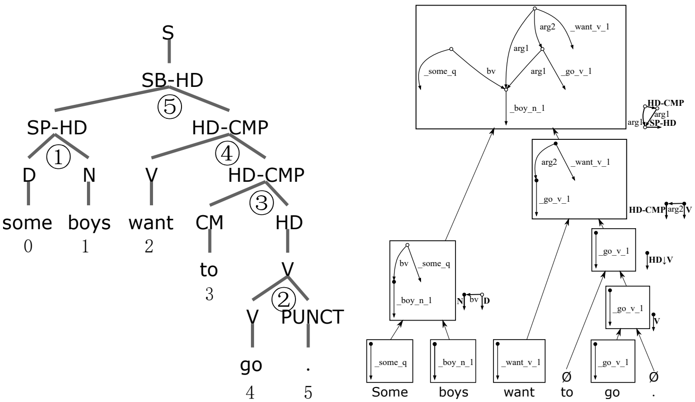


Synchronous Hyperedge Replacement Grammar



	①	②	③	④	⑤
Shared LHS	SP-HD	HD↓V	HD-CMP	HD-CMP	S↓SP-HD
RHS (syntax)	D + N	V + PUNCT	CM + HD↓V	V + HD-CMP	SP-HD + HD-CMP
RHS (semantics)	N bv D	V	HD↓V	HD-CMP arg2 V	HD-CMP arg1 arg1 SP-HD

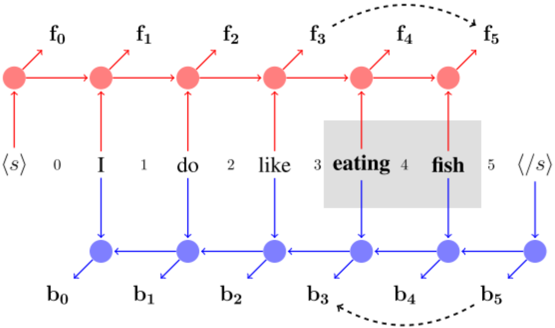
A Neural SHRG Parser



(a) Syntactic Parsing (b) Semantic Interpretation

Syntactic Parsing

- An LSTM-Minus-based constituent parser: use the difference of the LSTM outputs of the begining and ending words as span embedding $\mathrm{s}_{i,j}$
- A CKY decoder: restrict the parser to follow CFG rules.



Semantic Interpretation: Local Models

- We assume that each HRG rule is selected independently from the others. The score of G is defined as the sum of all rule scores:

$$\textsc{Score}(R = \{r_1, r_2, ...\}|T) = \sum_{r \in R} \textsc{Score}(r|T)$$

- The simplest count-based model: the rule score is estimated by its frequency in the training data

$$\textsc{Score}(r|T) = f(r)$$

Semantic Interpretation: Local Models

- Rule-Embedding-Based neural model: the rule score is estimated by neural network with span embedding $\mathsf{s}_{i,j}$ and rule embedding r:

$$\textsc{Score}(r|T) = \mathrm{MLP}(\mathsf{s}_{i,j} \oplus \mathsf{r})$$

- Inspired by the bag-of-words model, we represent the rule as bag of edge labels The i-th position in r indicates the number of times the i-th label appears in the rule

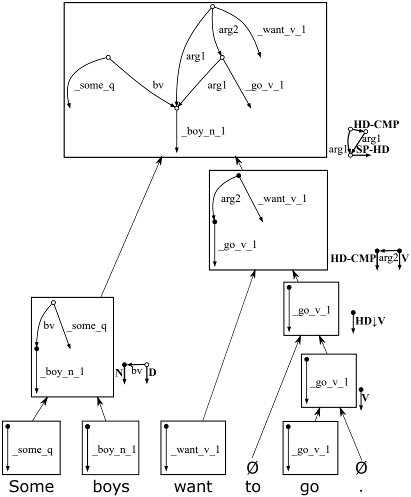
Semantic Interpretation: Structured Prediction

- Factorize $\textsc{Score}(R|T)$
- Assign scores to the graph and subgraphs in the intermediate state. Use predicates and arguments as factors for scoring

$$\textsc{Score}(R|T) = \sum_{i \in \textsc{Part}(R,T)} \textsc{ScorePart}(i)$$

- Perform the beam search in the bottom-up direction and only reserve top k subgraphs in each beam

The Bottom-Up Beam Search Process



Evaluation

- English Resource Grammar
- EDS/DMRS graphs
- DeepBank annotations

	Model	EDM_P	EDM_A	EDM
EDS	Buys, et al. 2017	88.14	82.20	85.48
	ACE (ERG)	91.82	86.92	89.58
	Ours (SHRG)	93.15	87.59	90.35
DMRS	Buys, et al. 2017	87.54	80.10	84.16
	ACE (ERG)	92.08	86.77	89.64
	Ours (SHRG)	93.11	86.01	89.51

Lessons learned

We think

- Explicit syntax-semantic interface is important, and SHRG is a good choice.
- Grammar formalism:
  Generative-enumerative vs. Model-theoretic

Lessons learned

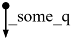
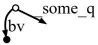
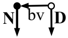
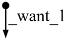
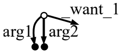
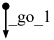
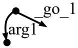
We think

- Explicit syntax-semantic interface is important, and SHRG is a good choice.
- Grammar formalism:
  Generative-enumerative vs. Model-theoretic

What's more

- From linguistics to computation
- From computation to linguistics

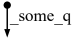
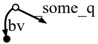
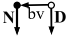
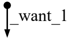
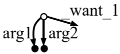
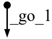
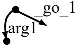
Lexicalism vs. Constructivism

Lexicon	Construction	Lexicalized	CFG Counterpart	Construction	Lexicalized
some			SP-HD→D+N		
want			HD-CMP→V+HD-CMP		
go			S↓SP-HD→SP-HD+HD-CMP		

Empirical evaluation results; I try to not interpretate it too much.

Grammar	EDM_P	EDM_A	EDM
Construction	93.48	87.88	90.67
Lexicalized	92.14	81.05	86.63

Lexicalism vs. Constructivism

Lexicon	Construction	Lexicalized	CFG Counterpart	Construction	Lexicalized
some			SP-HD→D+N		
want			HD-CMP→V+HD-CMP		
go			S↓SP-HD→SP-HD+HD-CMP		

Is it due to the sparseness?

Grammar	1	2	3	4	5+
Construction	14234	3424	1486	732	418
Lexicalized	11653	5938	2358	396	11

Outline

SHRG-based Parsing

Generation via DAG Transduction

Bi-directional Grammar

Linguistic performance

- Comprehension
- Production

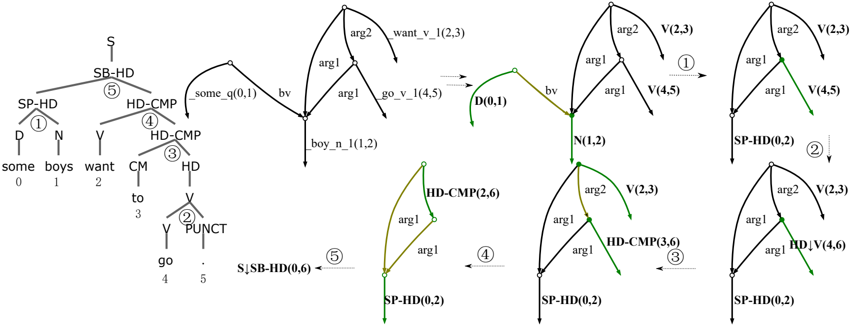
Many (computational) linguists assume bi-directional (competence) grammar, which we use for both comprehension and production.

E.g. DELPHIN's grammar family

Natural Language Processing

- Semantic parsing
- Surface realization

A synchronous grammar is naturally bi-directional



	①	②	③	④	⑤
Shared LHS	SP-HD	HD↓V	HD-CMP	HD-CMP	S↓SP-HD
RHS (syntax)	D + N	V + PUNCT	CM + HD↓V	V + HD-CMP	SP-HD + HD-CMP
RHS (semantics)	N bv D	V	HD↓V	HD-CMP arg2 V	HD-CMP arg1 arg1 SP-HD

SHRG-based surface generation

There is supposed to be some numbers about SHRG-based graph-to-string mapping, or say generation

SHRG-based surface generation

There is supposed to be some numbers about SHRG-based graph-to-string mapping, or say generation

SHRG-based surface generation

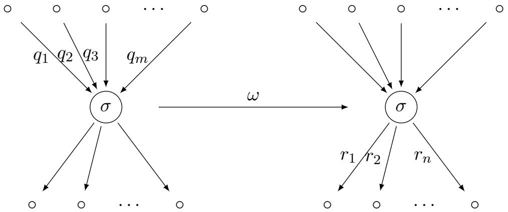
There is supposed to be some numbers about SHRG-based graph-to-string mapping, or say generation

Now another story.

DAG Automata

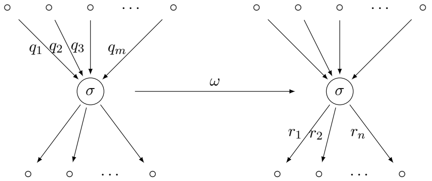
A weighted DAG automata is a tuple

$$M = \langle \Sigma, Q, \delta, \mathbb{K} \rangle$$



$$\{q_1, \cdots, q_m\} \xrightarrow{\sigma} \{r_1, \cdots, r_n\}$$

DAG Automata



- A run of M on DAG $D = \langle V, E, \ell \rangle$ is an edge labeling function $\rho : E \to Q$.
- The weight of ρ is the product of all weight of local transitions:

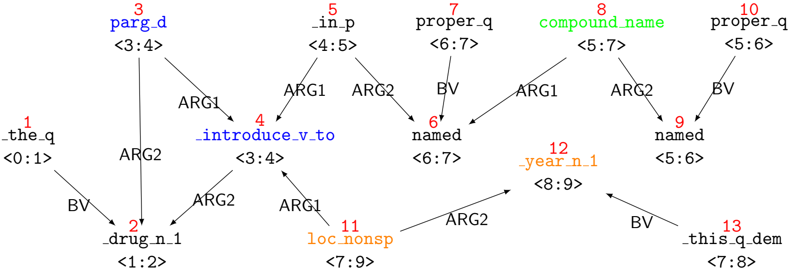
$$\delta(\rho) = \bigotimes_{v \in V} \delta \left[\rho(in(v)) \xrightarrow{\ell(v)} \rho(out(v)) \right]$$

Reference

David Chiang, Frank Drewes, Daniel Gildea, Adam Lopez and Giorgio Satta. Weighted DAG Automata for Semantic Graphs.

Type-Logical Semantic Graph

When we transform a graph grounded under type-logical semantics, usually we get a very *flat* graph.

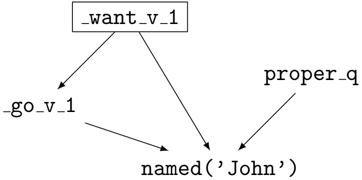


Challenges for DAG-to-tree transduction

- Cannot reverse the directions of edges
- Cannot handle multiple roots

Our transducer

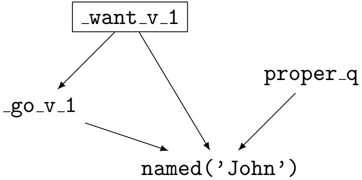
- The basic idea is to give up the rewritting way to directly generate a new data structure piece by piece, during recognizing an input DAG.
- Instead, our transducer obtains target structures based on side effects of DAG recognition.

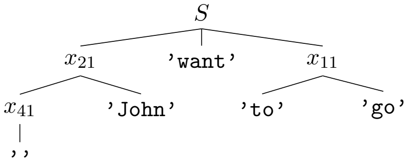


$$S = x_{21} + \texttt{'want'} + x_{11}$$
$$x_{11} = \texttt{'to'} + \texttt{'go'}$$
$$x_{21} = x_{41} + \texttt{'John'}$$
$$x_{41} = \texttt{''}$$

Our transducer

- The basic idea is to give up the rewritting way to directly generate a new data structure piece by piece, during recognizing an input DAG.
- Instead, our transducer obtains target structures based on side effects of DAG recognition.





Our DAG-to-program transducer

$Q = \{\texttt{DET(1,'r')}, \texttt{Empty(0,'e')}, \texttt{VP(1,'n')}, \texttt{NP(1,'n')}\}$		
Rule	For Recognition	For Generation
1	$\{\} \xrightarrow{\texttt{proper\_q}} \{\texttt{DET(1,'r')}\}$	$v_{\texttt{DET(1,'r')}} = \texttt{''}$
2	$\{\} \xrightarrow{\texttt{\_want\_v\_1}} \{\texttt{VP(1,'n')}, \texttt{NP(1,'n')}\}$	$S = v_{\texttt{NP(1,'n')}} + L + v_{\texttt{VP(1,'n')}}$
3	$\{\texttt{VP(1,'n')}\} \xrightarrow{\texttt{\_go\_v\_1}} \{\texttt{Empty(0,'e')}\}$	$v_{\texttt{VP(1,'n')}} = \texttt{'to'} + L$
4	$\{\texttt{NP(1,'n')}, \texttt{DET(1,'r')}\} \xrightarrow{\texttt{named}} \{\}$	$v_{\texttt{NP(1,'n')}} = v_{\texttt{DET(1,'r')}} + L$

- Every state q is of the form $label(num, \texttt{'n'} | \texttt{'e'} | \texttt{'r'})$

Our DAG-to-program transducer

$Q = \{\texttt{DET(1,'r')}, \texttt{Empty(0,'e')}, \texttt{VP(1,'n')}, \texttt{NP(1,'n')}\}$

Rule	For Recognition	For Generation
1	$\{\} \xrightarrow{\texttt{proper\_q}} \{\texttt{DET(1,'r')}\}$	$v_{\texttt{DET(1,'r')}} = \texttt{''}$
2	$\{\} \xrightarrow{\texttt{\_want\_v\_1}} \{\texttt{VP(1,'n')}, \texttt{NP(1,'n')}\}$	$S = v_{\texttt{NP(1,'n')}} + L + v_{\texttt{VP(1,'n')}}$
3	$\{\texttt{VP(1,'n')}\} \xrightarrow{\texttt{\_go\_v\_1}} \{\texttt{Empty(0,'e')}\}$	$v_{\texttt{VP(1,'n')}} = \texttt{'to'} + L$
4	$\{\texttt{NP(1,'n')}, \texttt{DET(1,'r')}\} \xrightarrow{\texttt{named}} \{\}$	$v_{\texttt{NP(1,'n')}} = v_{\texttt{DET(1,'r')}} + L$

- Every state q is of the form $label(num, \texttt{'n'|'e'|'r'})$
- $v_{\texttt{XXX(2,'r')}}$ means the second variable of state `XXX(3,'r')`

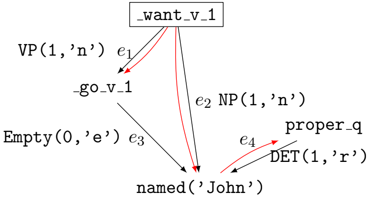
Our DAG-to-program transducer

$Q = \{\texttt{DET(1,'r')}, \texttt{Empty(0,'e')}, \texttt{VP(1,'n')}, \texttt{NP(1,'n')}\}$		
Rule	For Recognition	For Generation
1	$\{\} \xrightarrow{\texttt{proper\_q}} \{\texttt{DET(1,'r')}\}$	$v_{\texttt{DET(1,'r')}} = \texttt{''}$
2	$\{\} \xrightarrow{\texttt{\_want\_v\_1}} \{\texttt{VP(1,'n')}, \texttt{NP(1,'n')}\}$	$S = v_{\texttt{NP(1,'n')}} + L + v_{\texttt{VP(1,'n')}}$
3	$\{\texttt{VP(1,'n')}\} \xrightarrow{\texttt{\_go\_v\_1}} \{\texttt{Empty(0,'e')}\}$	$v_{\texttt{VP(1,'n')}} = \texttt{'to'} + L$
4	$\{\texttt{NP(1,'n')}, \texttt{DET(1,'r')}\} \xrightarrow{\texttt{named}} \{\}$	$v_{\texttt{NP(1,'n')}} = v_{\texttt{DET(1,'r')}} + L$

- Every state q is of the form $label(num, \texttt{'n'|'e'|'r'})$
- $v_{\texttt{XXX(2,'r')}}$ means the second variable of state `XXX(3,'r')`
- $S \rightarrow$ whole sentence
- $L \rightarrow$ output string of current node label

Our DAG-to-program transducer

$Q = \{\texttt{DET(1,'r')}, \texttt{Empty(0,'e')}, \texttt{VP(1,'n')}, \texttt{NP(1,'n')}\}$		
Rule	For Recognition	For Generation
1	$\{\} \xrightarrow{\texttt{proper\_q}} \{\texttt{DET(1,'r')}\}$	$v_{\texttt{DET(1,'r')}} = \texttt{''}$
2	$\{\} \xrightarrow{\texttt{\_want\_v\_1}} \{\texttt{VP(1,'n')}, \texttt{NP(1,'n')}\}$	$S = v_{\texttt{NP(1,'n')}} + L + v_{\texttt{VP(1,'n')}}$
3	$\{\texttt{VP(1,'n')}\} \xrightarrow{\texttt{\_go\_v\_1}} \{\texttt{Empty(0,'e')}\}$	$v_{\texttt{VP(1,'n')}} = \texttt{'to'} + L$
4	$\{\texttt{NP(1,'n')}, \texttt{DET(1,'r')}\} \xrightarrow{\texttt{named}} \{\}$	$v_{\texttt{NP(1,'n')}} = v_{\texttt{DET(1,'r')}} + L$



Recognition: To find an edge labeling function ρ. The red edges in make up an intermediate graph $T(\rho)$

Our DAG-to-program transducer

$Q = \{\texttt{DET(1,'r')}, \texttt{Empty(0,'e')}, \texttt{VP(1,'n')}, \texttt{NP(1,'n')}\}$		
Rule	For Recognition	For Generation
1	$\{\} \xrightarrow{\texttt{proper\_q}} \{\texttt{DET(1,'r')}\}$	$v_{\texttt{DET(1,'r')}} = \texttt{''}$
2	$\{\} \xrightarrow{\texttt{\_want\_v\_1}} \{\texttt{VP(1,'n')}, \texttt{NP(1,'n')}\}$	$S = v_{\texttt{NP(1,'n')}} + L + v_{\texttt{VP(1,'n')}}$
3	$\{\texttt{VP(1,'n')}\} \xrightarrow{\texttt{\_go\_v\_1}} \{\texttt{Empty(0,'e')}\}$	$v_{\texttt{VP(1,'n')}} = \texttt{'to'} + L$
4	$\{\texttt{NP(1,'n')}, \texttt{DET(1,'r')}\} \xrightarrow{\texttt{named}} \{\}$	$v_{\texttt{NP(1,'n')}} = v_{\texttt{DET(1,'r')}} + L$

$$S = x_{21} + \texttt{'want'} + x_{11}$$

$$x_{11} = \texttt{'to'} + \texttt{'go'}$$

$$x_{21} = x_{41} + \texttt{'John'}$$

$$x_{41} = \texttt{''}$$

Instantiation: replace $v_{l(j,d)}$ of edge e_i with variable x_{ij} and L with the output string.

DAG Transduction based-NLG

A general framework for DAG transduction based-NLG:

First phase: Syntax

Use a DAG transducer to translate a semantic graph into sequential lemmas

Second phase: Morphology

Use a neural sequence-to-sequence model to obtain final surface strings from lemmas

Fine-to-coarse transduction

A Fine-to-Coarse strategy is used to ensure that at least one sentence is generated for any input graph.

- induced-rules (for precision)
- extended-rules
- dynamic rules (for robustness)

During decoding, when neither induced nor extended rule is applicable, we use a Markov model to *create* a dynamic rule on-the-fly:

$$P(\{q_1, \cdots, q_n\}|C) = P(q_1|C)\prod_{i=2}^{n} P(q_i|C)P(q_i|q_{i-1}, C)$$

NLG via DAG transduction

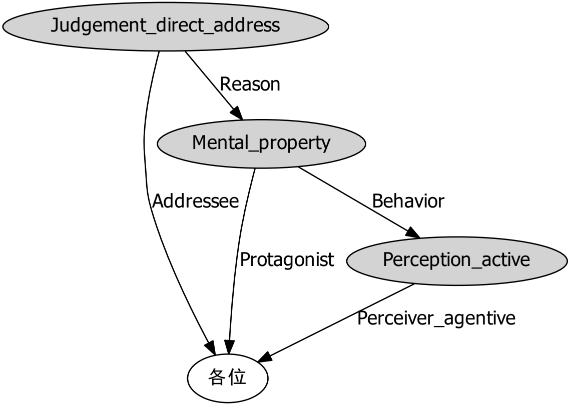
Experimental set-up

- Data: DeepBank + Wikiwoods
- Decoder: Beam search (beam size = 128)
- Other tool: OpenNMT

Transducer	Lemmas	Sentences	Coverage
I	89.44	74.94	67%
I+E	88.41	74.03	77%
I+E+D	82.04	68.07	100%
DFS-NN	50.45		100%
AMR-NN		33.8	100%
AMR-NRG		25.62	100%

Conclusion and discussion

- The relevance of linguistic structures in neural natural language processing
- Formalisms help!
- *Deep* learning models help DEEP LINGUISTIC PROCESSING
- Beyond building practical NLP systems, computation helps evaluate a linguistic theory.


Judgement_direct_address
Reason
Mental_property
Addressee
Behavior
Protagonist
Perception_active
Perceiver_agentive
各位

感谢各位耐心聆听