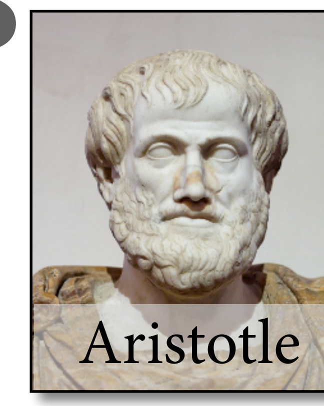
Semantic parsing

Computational Linguistics

Alexander Koller

28 January 2020

Computing with meanings



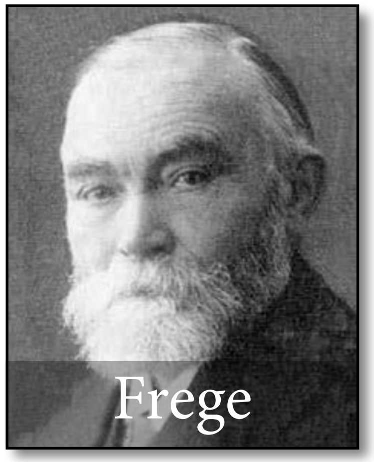
- Ancient problem: *inference*.
  - How can we tell whether a sentence follows from others?
  - Can we compute this automatically?

All men are mortal.

Socrates is a man.

Therefore, Socrates is mortal.

Formal meaning representations



- Aristotle with more modern tools (ca. 2000):
  - Compute *meaning representation* in some formal language (e.g. predicate logic)
  - so that it captures something relevant about the sentence's meaning (e.g. its *truth conditions*)
  - and then use reasoning tools for the formal language (e.g. a *theorem prover* for predicate logic)

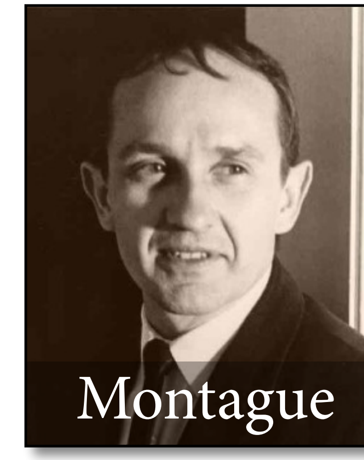
All men are mortal.
Socrates is a man.

Therefore, Socrates is mortal.

$\forall x.\ \mathrm{man}(x) \rightarrow \mathrm{mortal}(x)$
$\mathrm{man}(s)$

$\mathrm{mortal}(s)$

Compositional semantics



$S \rightarrow NP\ VP$	$\langle S \rangle = \langle NP \rangle(\langle VP \rangle)$
$VP \rightarrow V\ NP$	$\langle VP \rangle = \lambda y\ \langle NP \rangle(\langle V \rangle(y))$
$NP \rightarrow Det\ N$	$\langle NP \rangle = \langle Det \rangle(\langle N \rangle)$
$NP \rightarrow$ John	$\langle NP \rangle = \lambda P\ P(j^*)$
$V \rightarrow$ eats	$\langle V \rangle = eat'$
$Det \rightarrow$ a	$\langle Det \rangle = \lambda P \lambda Q \exists x\ P(x) \wedge Q(x)$
$N \rightarrow$ sandwich	$\langle N \rangle = sw'$

when you apply this syntax rule …

… construct λ-term for parent from λ-terms for children like this

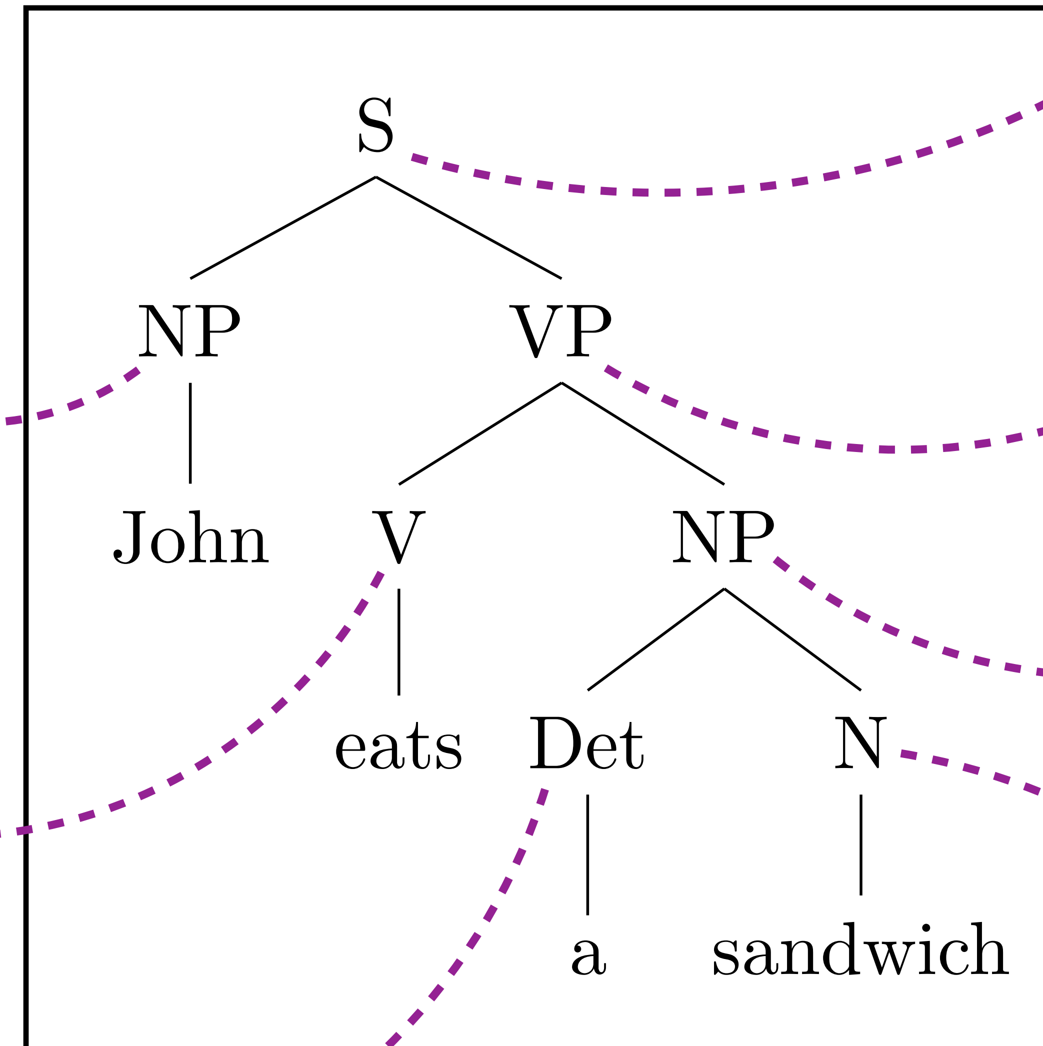
Example

$(\lambda P\ P(j^*))\ (\lambda y\ \exists x\ sw'(x) \wedge eat'(y)(x))$

$\rightarrow_\beta (\lambda y\ \exists x\ sw'(x) \wedge eat'(y)(x))(j^*)$

$\rightarrow_\beta \exists x\ sw'(x) \wedge eat'(j^*)(x)$

$\lambda P\ P(j^*)$



$\lambda y\ (\lambda Q \exists x\ sw'(x) \wedge Q(x))(eat'(y))$

$\rightarrow_\beta \lambda y\ \exists x\ sw'(x) \wedge eat'(y)(x)$

$(\lambda P \lambda Q \exists x\ P(x) \wedge Q(x))(sw')$

$\rightarrow_\beta \lambda Q \exists x\ sw'(x) \wedge Q(x)$

eat'

sw'

$\lambda P \lambda Q \exists x\ P(x) \wedge Q(x)$

Semantic parsing

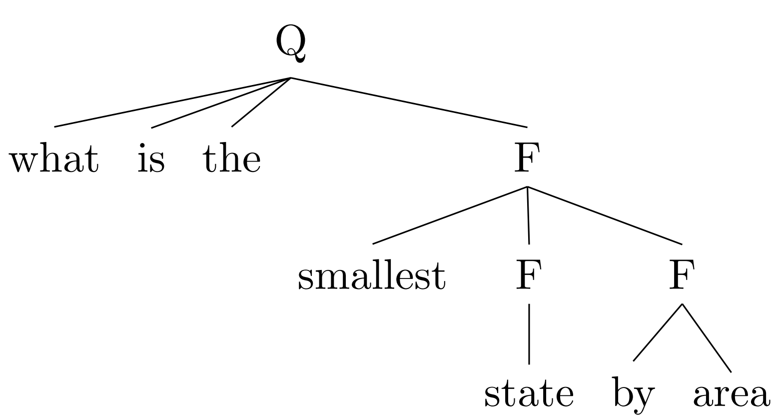
- Open issue in classical semantics construction: Where do we get large grammar that supports it?
- Current trend in CL is *semantic parsing*: learn mapping from sentence to formal meaning representation using statistical methods.
- E.g. from Geoquery corpus (880 sentences):

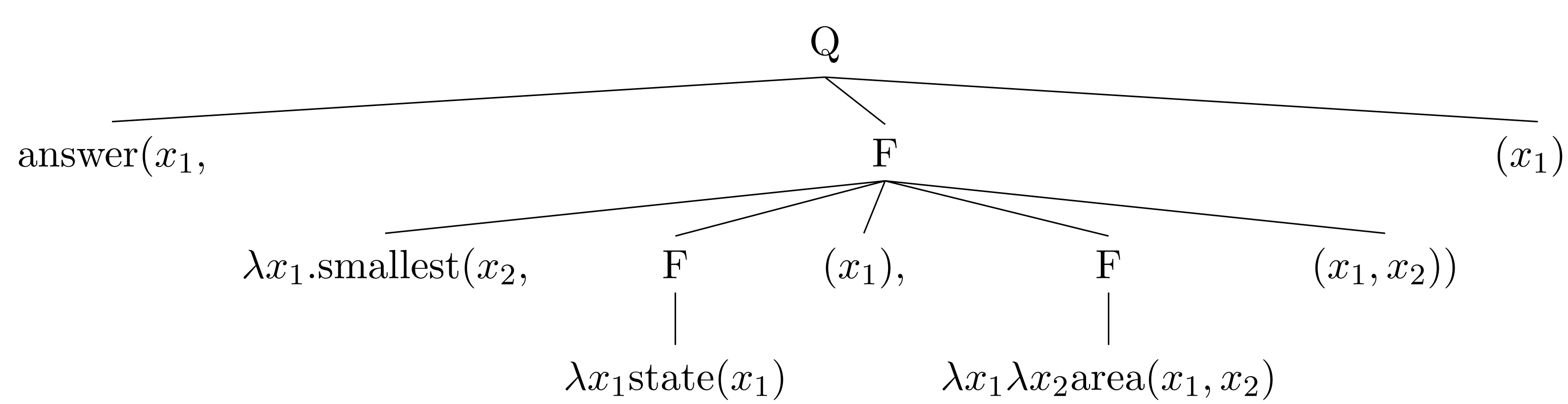
> What is the smallest state by area?
>
> `answer(x1, smallest(x2, state(x1), area(x1, x2)))`

With synchronous grammars

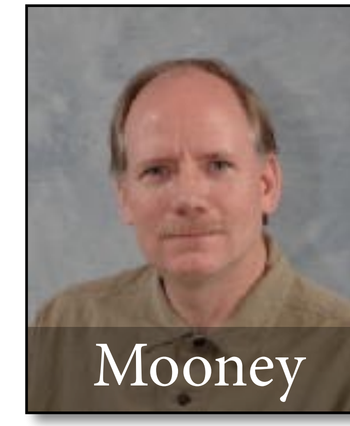
- Use a synchronous grammar (≈ SCFG) to simultaneously generate strings and λ-expressions.

Q → what is the F	Q → answer(x_1, F(x_1))
F → smallest F F	F → λx_1 smallest(x_2, F(x_1), F(x_1, x_2))
F → state	F → λx_1 state(x_1)
F → by area	F → λx_1 λx_2 area(x_1, x_2)



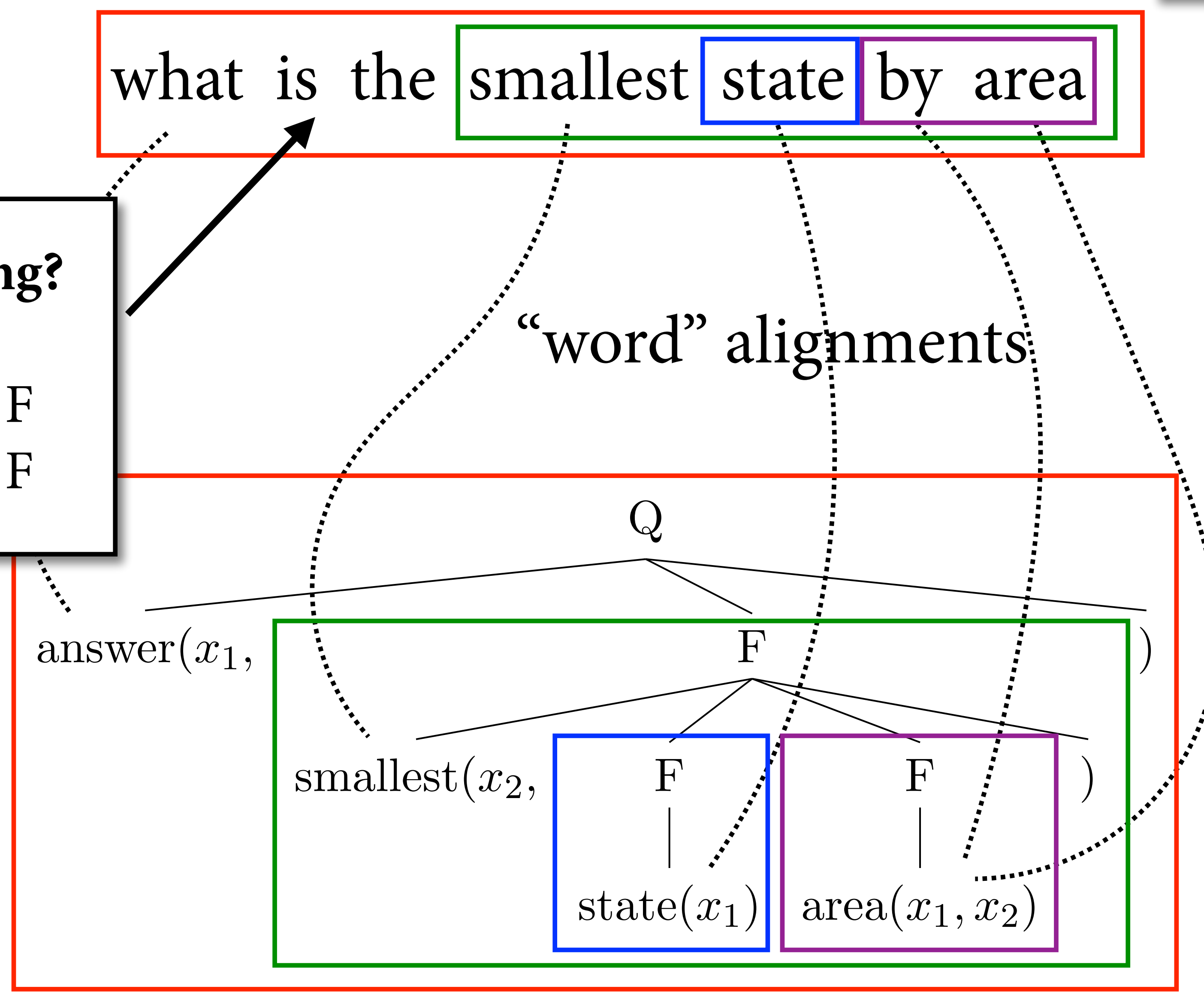


Wong & Mooney



Where do unaligned words belong?

Q → what is the F | F → smallest F
Q → what F | F → is the smallest F



Assumptions:
- alignments between words and nodes
- unambiguous structure of meaning representation

Combinatory categorial grammar

John	eats	a	big	sandwich	
NP	(S\NP)/NP	NP/N	N/N	N	
			N		>
		NP			>
	S\NP				>
S					<

Semantics in CCG

$$\frac{X\text{: } a}{Y/(Y\backslash X)\text{: } \lambda P.P(a)}{>}T \qquad \frac{X/Y\text{: } f \qquad Y/Z\text{: } g}{X/Z\text{: } \lambda x.f(g(x))}{>}B \qquad \frac{X/Y\text{: } f \qquad Y\backslash Z\text{: } g}{X\backslash Z\text{: } \lambda x.f(g(x))}{>}Bx$$

$$\frac{X\text{: } a}{Y\backslash(Y/X)\text{: } \lambda P.P(a)}{<}T \qquad \frac{Y\backslash Z\text{: } g \qquad X\backslash Y\text{: } f}{X\backslash Z\text{: } \lambda x.f(g(x))}{<}B \qquad \frac{Y/Z\text{: } g \qquad X\backslash Y\text{: } f}{X/Z\text{: } \lambda x.f(g(x))}{<}Bx$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{John}}{NP\text{: } h^*}}{S/(S\backslash NP)\text{: } \lambda P.P(h^*)}{>}T \qquad \dfrac{\text{eats}}{(S\backslash NP)/NP\text{: } eat'}}{S/NP\text{: } \lambda x.(\lambda P.P(h^*))(eat'(x)) \Rightarrow_\beta \lambda x.eat'(x)(h^*)}{>}B \qquad \dfrac{\text{a sandwich}}{NP\text{: } sw'}}{S\text{: } (\lambda x.eat'(x)(h^*))(sw') \Rightarrow_\beta eat'(sw')(h^*)}{>}$$

Zettlemoyer & Collins

GENLEX: build candidates for lexicon entries

Rules		Categories produced from logical form
Input Trigger	Output Category	$\arg\max(\lambda x.state(x) \wedge borders(x, texas), \lambda x.size(x))$
constant c	$NP : c$	$NP : texas$
arity one predicate p_1	$N : \lambda x.p_1(x)$	$N : \lambda x.state(x)$
arity one predicate p_1	$S\backslash NP : \lambda x.p_1(x)$	$S\backslash NP : \lambda x.state(x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(y, x)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(y, x)$
arity two predicate p_2	$(S\backslash NP)/NP : \lambda x.\lambda y.p_2(x, y)$	$(S\backslash NP)/NP : \lambda x.\lambda y.borders(x, y)$
arity one predicate p_1	$N/N : \lambda g.\lambda x.p_1(x) \wedge g(x)$	$N/N : \lambda g.\lambda x.state(x) \wedge g(x)$
literal with arity two predicate p_2 and constant second argument c	$N/N : \lambda g.\lambda x.p_2(x, c) \wedge g(x)$	$N/N : \lambda g.\lambda x.borders(x, texas) \wedge g(x)$
arity two predicate p_2	$(N\backslash N)/NP : \lambda x.\lambda g.\lambda y.p_2(x, y) \wedge g(x)$	$(N\backslash N)/NP : \lambda g.\lambda x.\lambda y.borders(x, y) \wedge g(x)$
an $\arg\max$ / $\min$ with second argument arity one function f	$NP/N : \lambda g. \arg\max / \min(g, \lambda x.f(x))$	$NP/N : \lambda g. \arg\max(g, \lambda x.size(x))$
an arity one numeric-ranged function f	$S/NP : \lambda x.f(x)$	$S/NP : \lambda x.size(x)$

Log-linear probability models

- Define probability of parse tree in terms of *features:*

$$P(t \mid w) = \frac{e^{\theta \cdot f(t,w)}}{\sum_{t'} e^{\theta \cdot f(t',w)}}$$

  where $\theta \cdot f(t,w) = \theta_1 \cdot f_1(t,w) + \ldots + \theta_n \cdot f_n(t,w)$

- Features f(t,w) can capture arbitrary properties of t and w.
  - Here: Each feature counts uses of one grammar rule.

- Train weight vector θ from data.

Zettlemoyer & Collins

overall learning algorithm

Algorithm:

- For $t = 1 \ldots T$

Step 1: (Lexical generation)

- For $i = 1 \ldots n$:
  - Set $\lambda = \Lambda_0 \cup \text{GENLEX}(S_i, L_i)$.
  - Calculate $\pi = \text{PARSE}(S_i, L_i, \lambda, \bar{\theta}^{t-1})$.
  - Define λ_i to be the set of lexical entries in π.
- Set $\Lambda_t = \Lambda_0 \cup \bigcup_{i=1}^{n} \lambda_i$

Step 2: (Parameter Estimation)

- Set $\bar{\theta}^t = \text{ESTIMATE}(\Lambda_t, E, \bar{\theta}^{t-1})$

Evaluation results

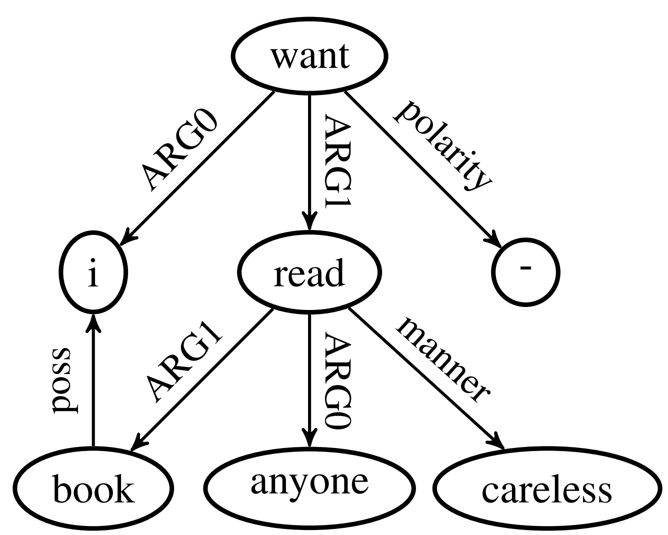
System	Variable Free			Lambda Calculus		
	Rec.	Pre.	F1	Rec.	Pre.	F1
Cross Validation Results						
KRISP	71.7	**93.3**	81.1	–	–	–
WASP	74.8	87.2	80.5	–	–	–
Lu08	81.5	89.3	**85.2**	–	–	–
λ-WASP	–	–	–	86.6	92.0	89.2
Independent Test Set						
ZC05	–	–	–	79.3	**96.3**	87.0
ZC07	–	–	–	86.1	91.6	88.8
UBL	81.4	89.4	**85.2**	85.0	94.1	**89.3**
UBL-s	**84.3**	85.2	84.7	**87.9**	88.5	88.2

(on Geoquery 880 corpus)

Abstract Meaning Representations

- Pros and cons of Geoquery:
  - semantic representations are trees — (too) easy
  - very small

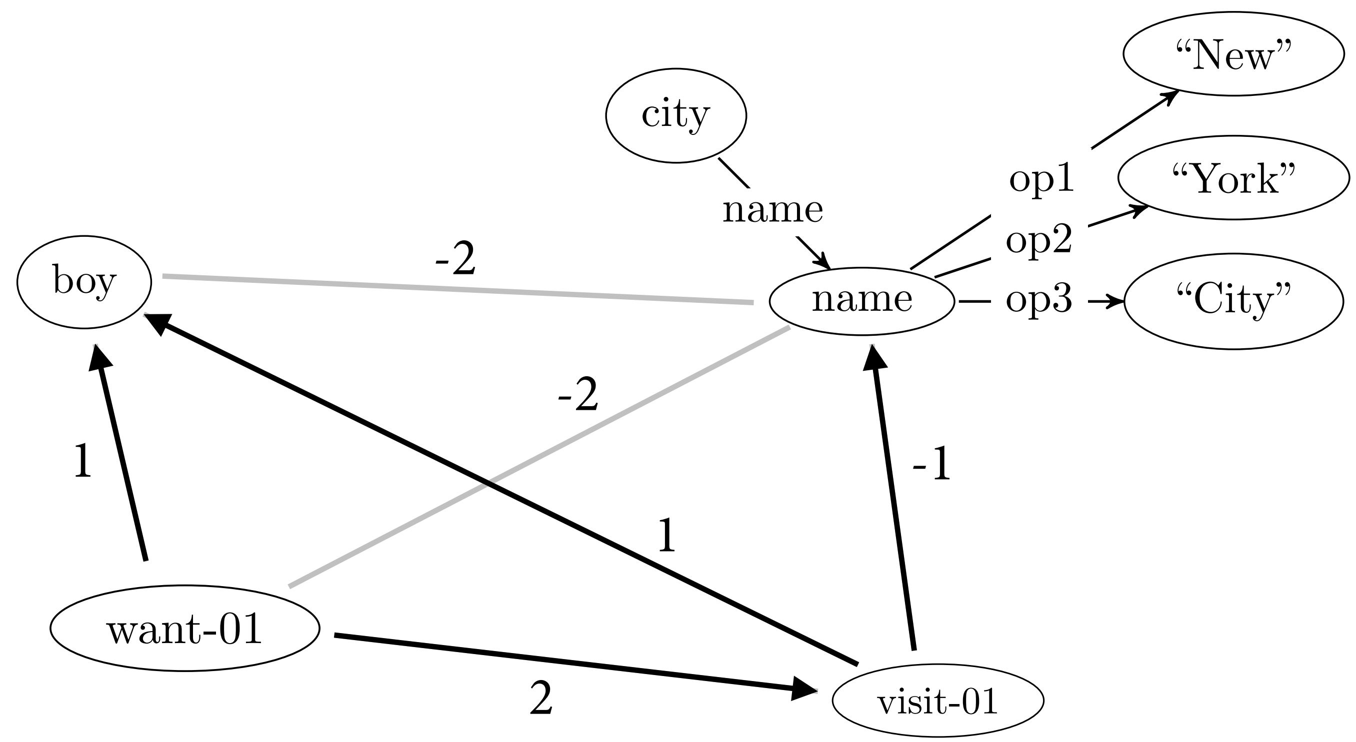
- Since 2014, much larger corpora available: ~40k AMRs, graphs as semantic representations.



"I don't want anyone to read my book carelessly."

Dependency-style AMR parsing

"The boy wants to visit New York City."



Concept Identification: determine atomic graph for each word.

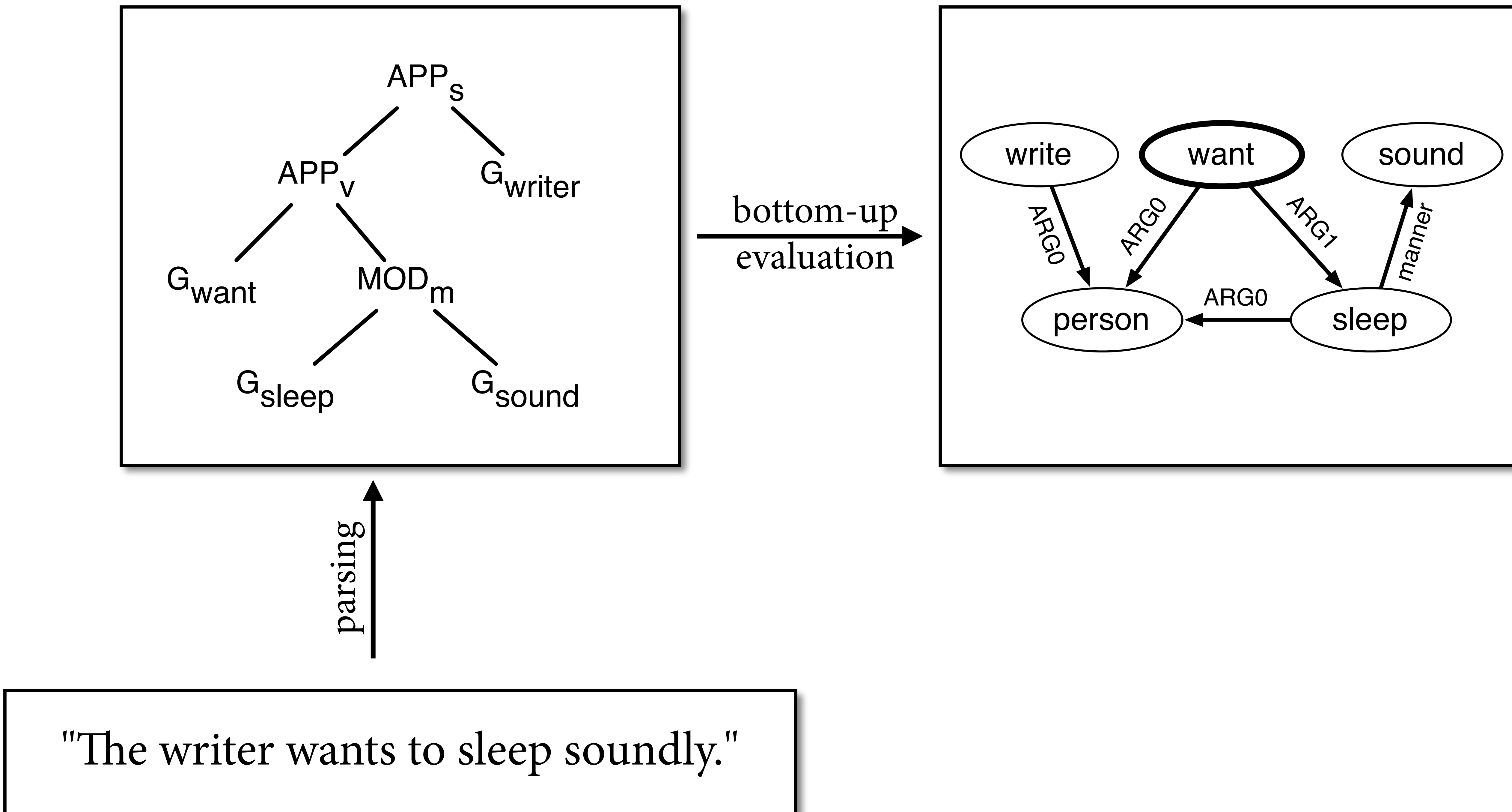
Relation Identification: add all edges with positive weight; then repeatedly add least negative edge that connects subgraphs.

JAMR; Flanigan et al. 2014

Issues with JAMR

- JAMR can draw edge between any two nodes; syntactic structure of sentence used only indirectly.
- Semantic representations for words don't know anything about their semantic arguments.
- Edges for control verbs added arbitrarily, not because linguistic principle of control discovered.
- No notion of compositionality!

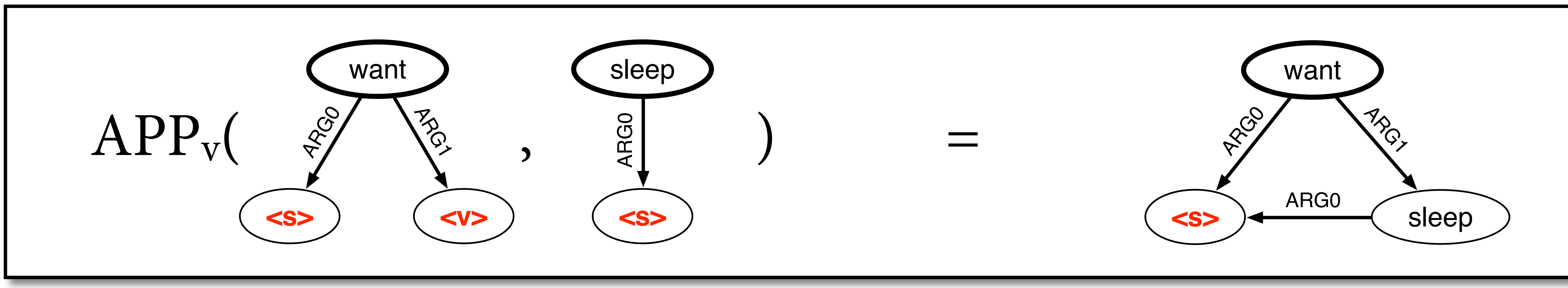
Compositional AMR Parsing

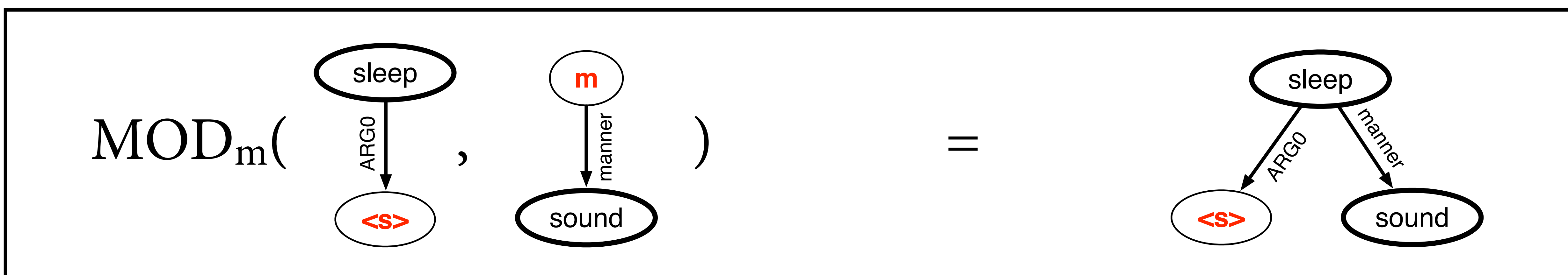


(Groschwitz et al., ACL 2018)

AM algebra

Two operations for combining s-graphs:
Apply (= head + complement), Modify (= head + modifier).

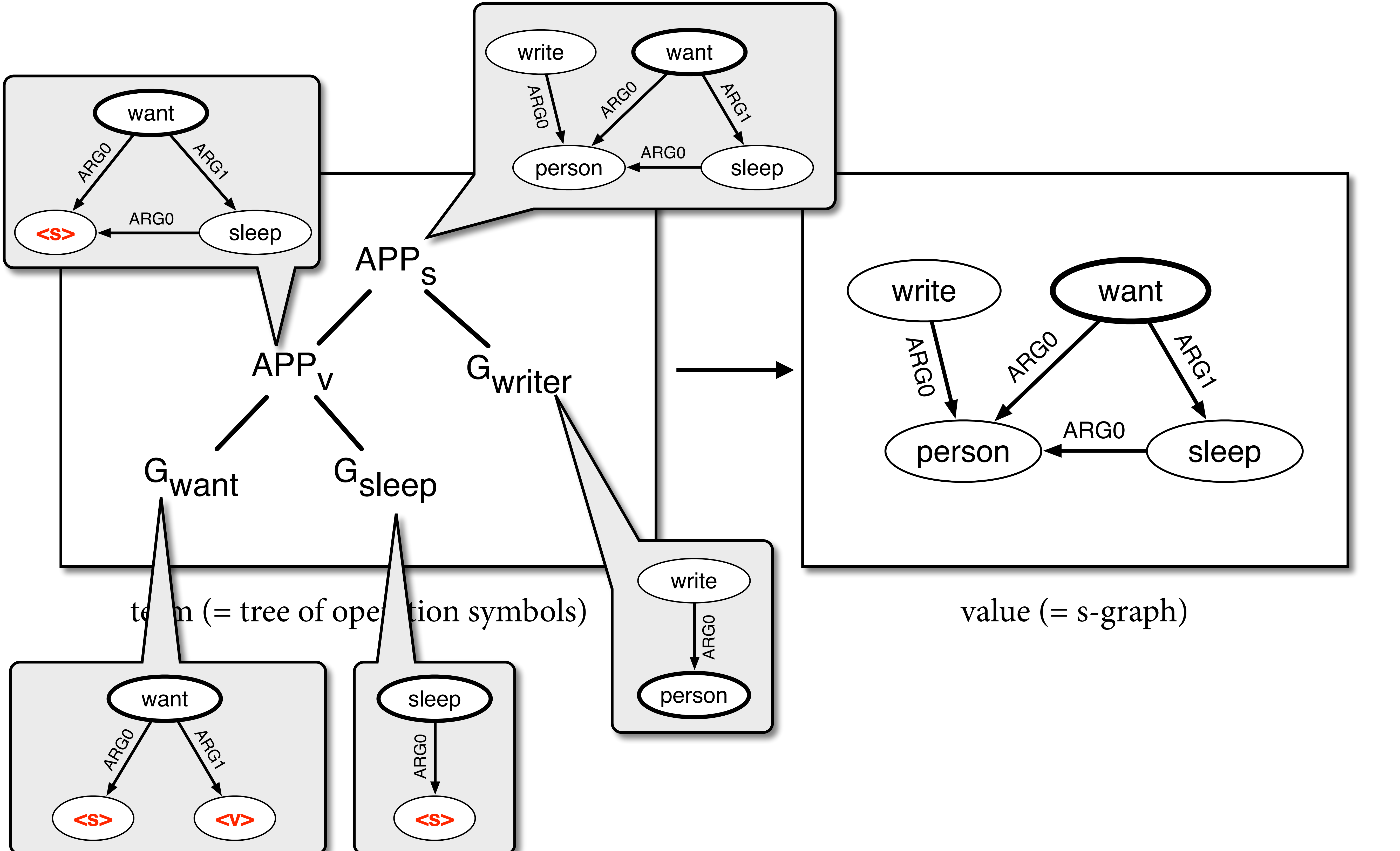




APP and MOD can be expressed in terms of rename, forget, merge.

(Groschwitz et al., IWCS 2017; inspired by Copestake et al. 2001)

AM terms



(Groschwitz et al., IWCS 2017)

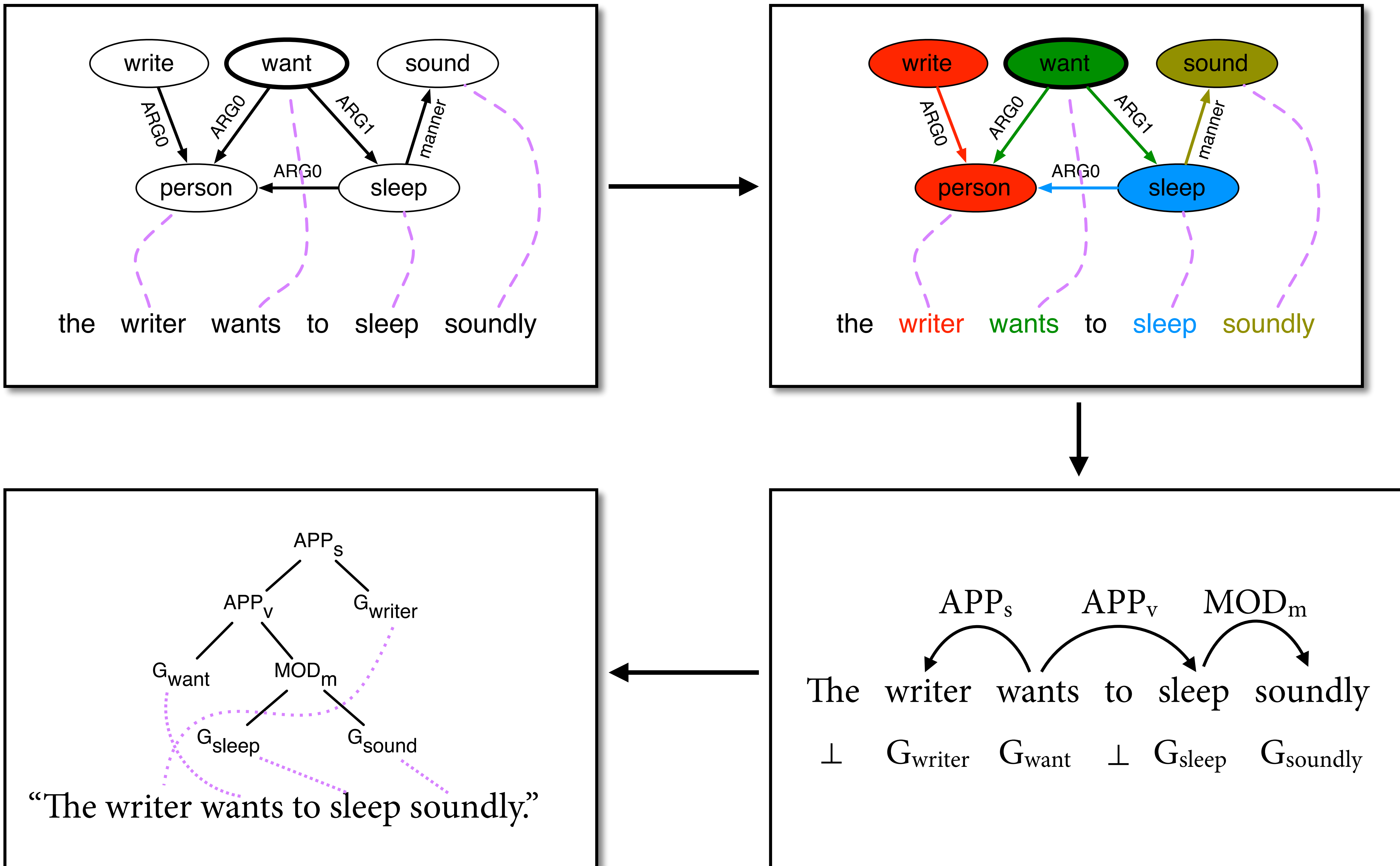
Approach

- Convert (string, graph) training data into (string, supertags + dependencies) training data.

- Train neural supertagger + dependency parser to assign scores to supertags + dependencies.
  - easier than predicting the whole graph; compositional!

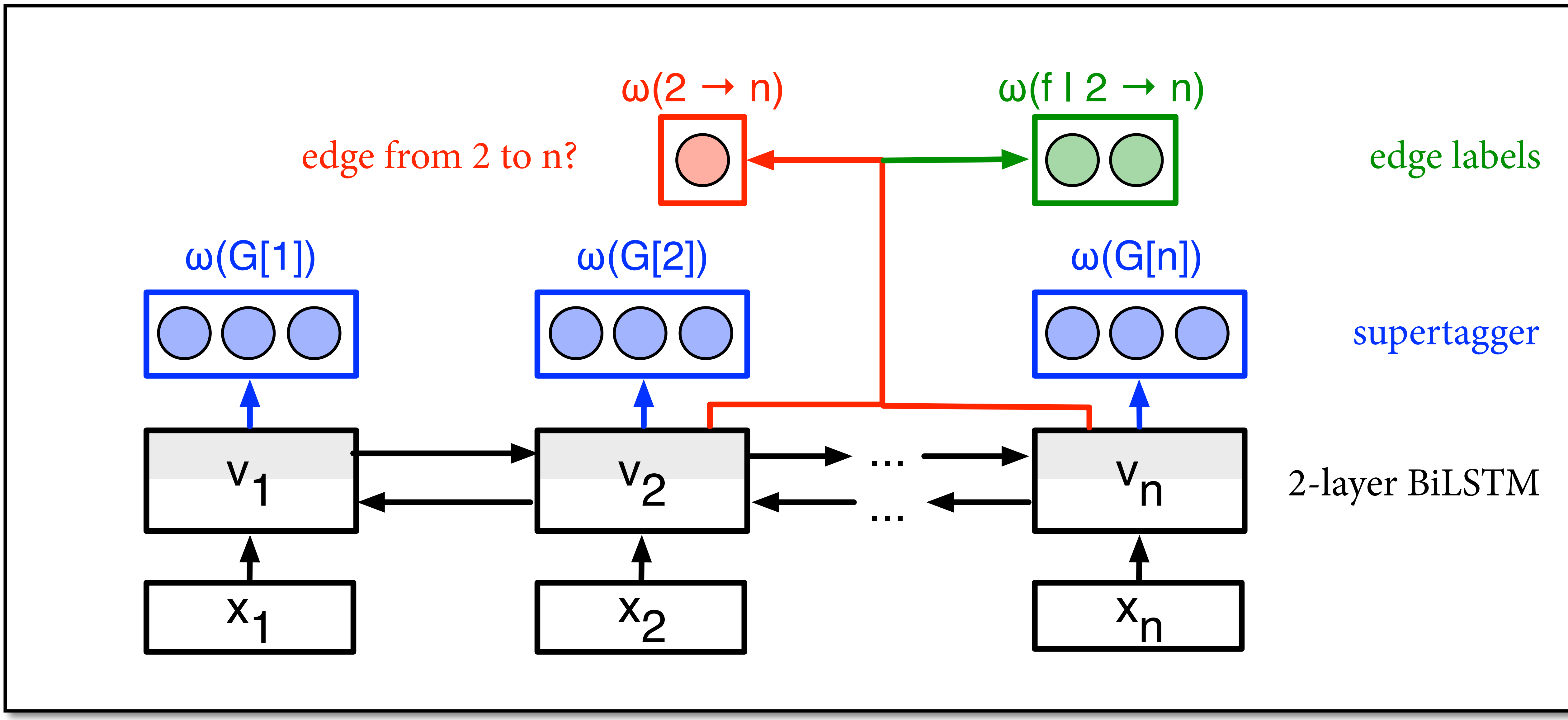
- At evaluation time, compute highest-scoring well-typed dependency tree.
  - well-typedness requirement makes this NP-complete
  - solve approximately with CKY-style parsing algorithm

Converting training data



(Groschwitz et al., ACL 2018)

Neural model



$\omega(2 \rightarrow n) = \log P(\text{edge from } 2 \rightarrow n \mid \mathbf{x})$ is *score* for this edge.
Analogously for supertags and edge labels.

(cf. Lewis et al. 2014; Kiperwasser & Goldberg 2016)

Parsing across graphbanks

	DM		PAS		PSD		EDS		AMR 2015	AMR 2017
	id F	ood F	id F	ood F	id F	ood F	Smatch F	EDM	Smatch F	Smatch F
Groschwitz et al. (2018)	-	-	-	-	-	-	-	-	70.2	71.0
Lyu and Titov (2018)	-	-	-	-	-	-	-	-	73.7	74.4 ±0.16
Zhang et al. (2019)	-	-	-	-	-	-	-	-	-	**76.3** ±0.1
Peng et al. (2017) Basic	89.4	84.5	92.2	88.3	77.6	75.3	-	-	-	-
Dozat and Manning (2018)	93.7	88.9	94.0	90.8	81.0	79.4	-	-	-	-
Buys and Blunsom (2017)	-	-	-	-	-	-	85.5	85.9	60.1	-
Chen et al. (2018)	-	-	-	-	-	-	**90.9**[1,2]	**90.4**[1]	-	-
This paper (GloVe)	90.4 ±0.2	84.3 ±0.2	91.4 ±0.1	86.6 ±0.1	78.1 ±0.2	74.5 ±0.2	87.6 ±0.1	82.5 ±0.1	69.2 ±0.4	70.7 ±0.2
This paper (BERT)	**93.9** ±0.1	**90.3** ±0.1	**94.5** ±0.1	**92.5** ±0.1	**82.0** ±0.1	**81.5** ±0.3	90.1 ±0.1	84.9 ±0.1	**74.3** ±0.2	75.3 ±0.2
Peng et al. (2017) Freda1	90.0	84.9	92.3	88.3	78.1	75.8	-	-	-	-
Peng et al. (2017) Freda3	90.4	85.3	92.7	89.0	78.5	76.4	-	-	-	-
This paper, MTL (GloVe)	91.2 ±0.1	85.7 ±0.0	92.2 ±0.2	88.0 ±0.3	78.9 ±0.3	76.2 ±0.4	88.2 ±0.1	83.3 ±0.1	(70.4)[3] ±0.2	71.2 ±0.2
This paper, MTL (BERT)	**94.1** ±0.1	**90.5** ±0.1	**94.7** ±0.1	**92.8** ±0.1	**82.1** ±0.2	**81.6** ±0.1	90.4 ±0.1	85.2 ±0.1	(74.5)[3] ±0.1	75.3 ±0.1

- First semantic parser that does well across all six major graphbanks.
- Established new states of the art through use of pretrained BERT embeddings.
- Small improvements through multi-task learning on multiple graphbanks.

(Lindemann et al., ACL 2019)

Conclusion

- Challenge in compositional semantic construction: Where do we get large-scale grammars?
- Semantic parsing: Learn such grammars from corpora with semantic annotations.
  - GeoQuery: small corpus of trees
  - AMRBank: new hotness
- Very active research topic right now.