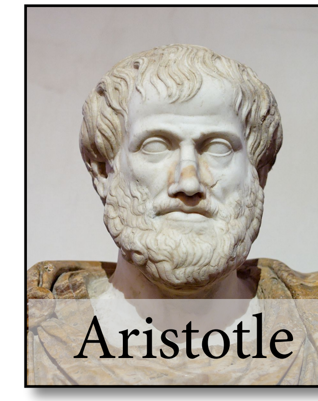
ANLP

18 - semantics

David Schlangen
University of Potsdam, MSc Cognitive Systems
Winter 2019 / 2020

Computing with meanings



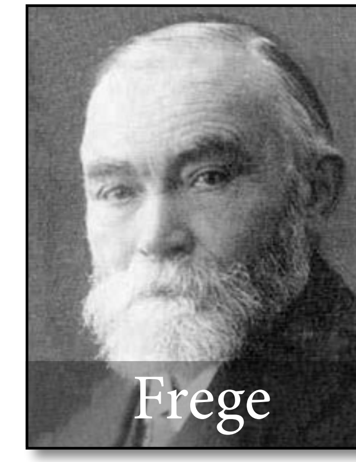
- Ancient problem: *inference.*
  - How can we tell whether a sentence follows from others?
  - Can we compute this automatically?

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)

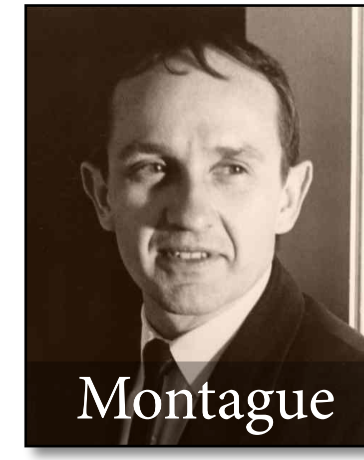
All men are mortal.
Socrates is a man.

Therefore, Socrates is mortal.

$\forall x.\ man(x) \rightarrow mortal(x)$
$man(s)$

$mortal(s)$

Compositional semantics



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

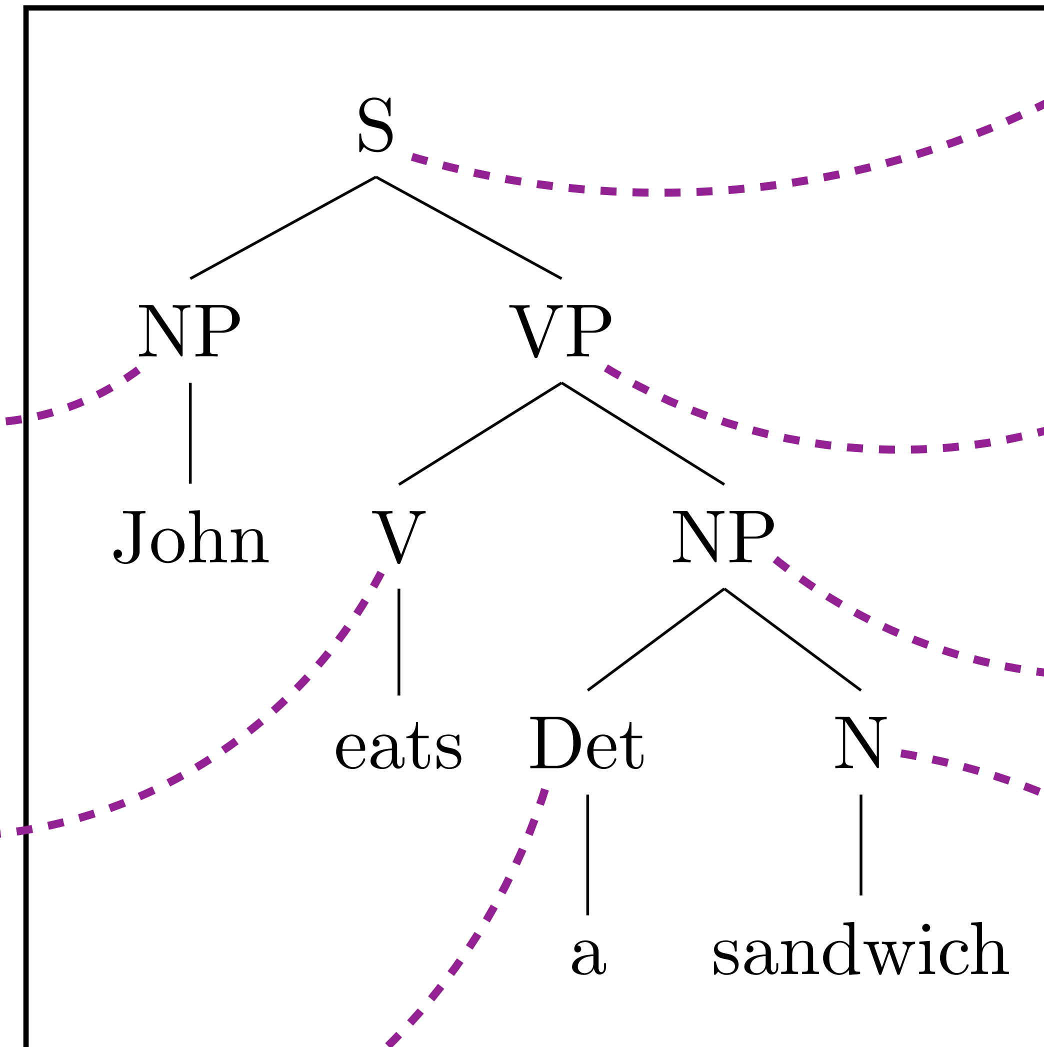
when you apply this syntax rule …

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

Example

(λP P(j’)) (λy ∃x sw’(x) ∧eat’(y)(x))
$\rightarrow_\beta$ (λy ∃x sw’(x) ∧eat’(y)(x))(j’)
$\rightarrow_\beta$ ∃x sw’(x) ∧eat’(j’)(x)

λP P(j’)



λy (λQ∃x sw’(x) ∧ Q(x))(eat’(y))
$\rightarrow_\beta$ λy ∃x sw’(x) ∧eat’(y)(x)

(λPλQ∃x P(x) ∧ Q(x))(sw’)
$\rightarrow_\beta$ λQ∃x sw’(x) ∧ Q(x)

eat’

sw’

λPλQ∃x P(x) ∧ 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.

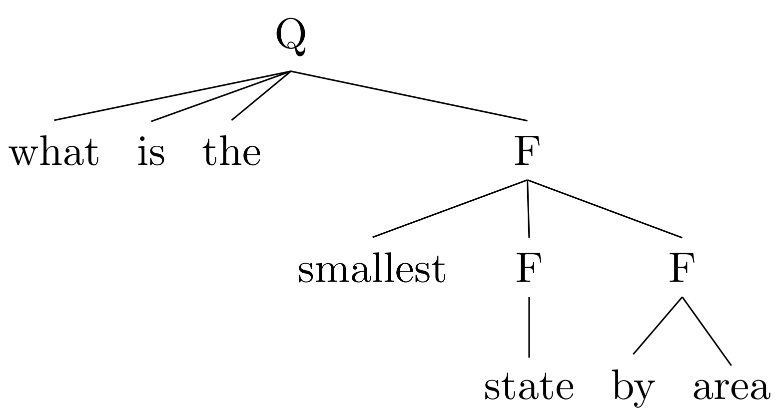
- E.g. from Geoquery corpus (880 sentences):

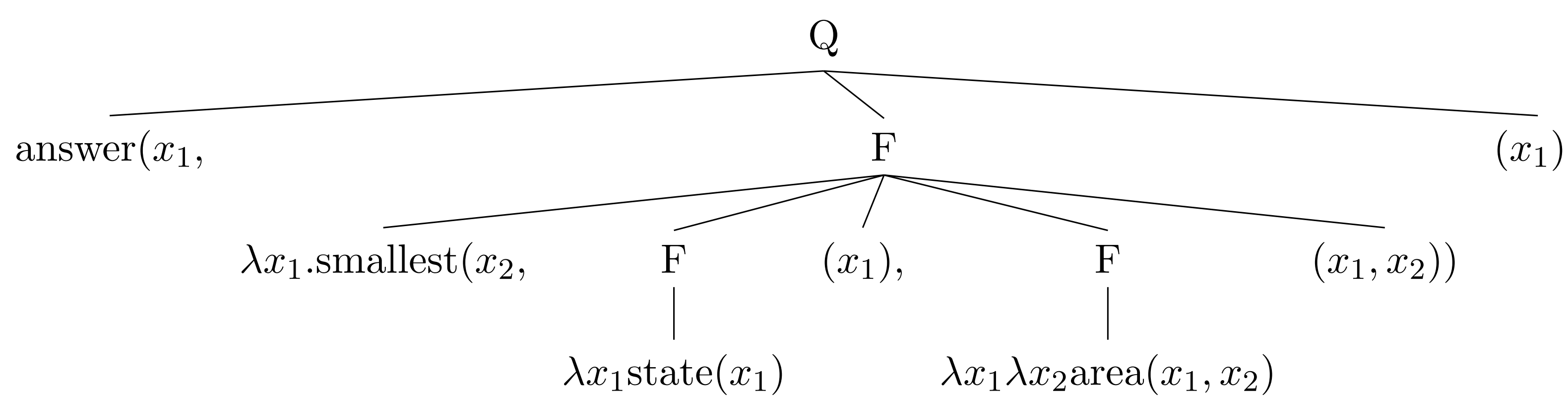
> What is the smallest state by area?
> `answer(x1, smallest(x2, state(x1), area(x1, x2)))`

With synchronous grammars

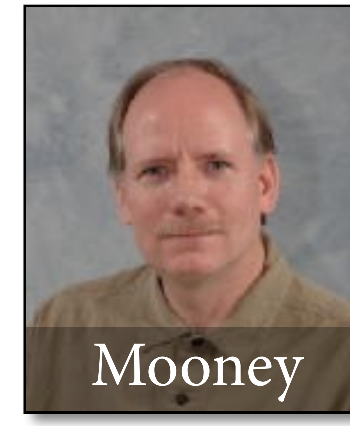
- Use a synchronous grammar (≈ SCFG) to simultaneously generate strings and λ-expressions.

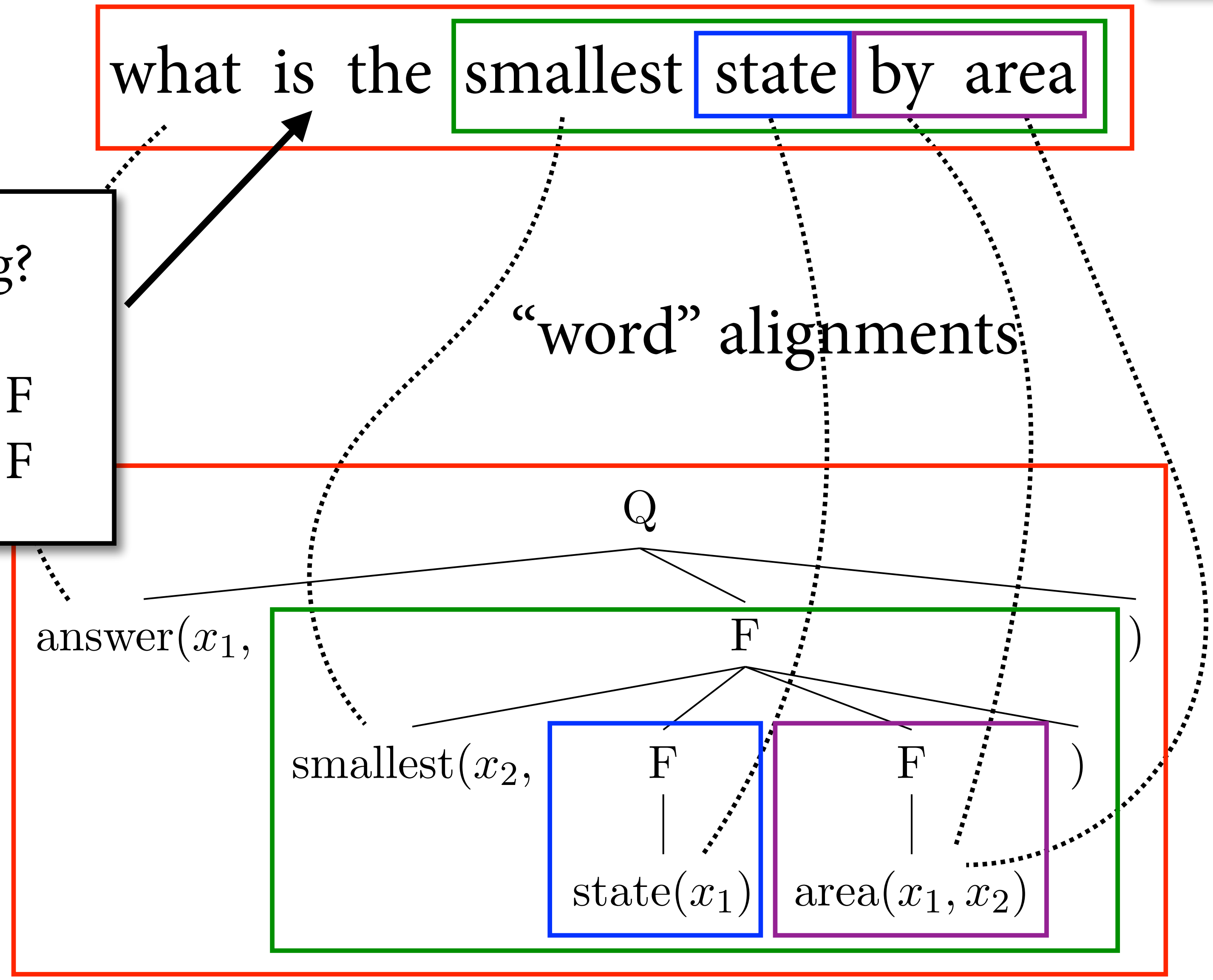
Q → what is the F	Q → answer(x_1, F(x_1))
F → smallest F F	F → λx_1 smallest(x_2, F(x_1), F(x_1, x_2))
F → state	F → λx_1 state(x_1)
F → by area	F → λx_1 λx_2 area(x_1, x_2)





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

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

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

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

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 $\lambda = \Lambda_0 \cup \mathrm{GENLEX}(S_i, L_i)$.
  - Calculate $\pi = \mathrm{PARSE}(S_i, L_i, \lambda, \bar{\theta}^{t-1})$.
  - 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 $\bar{\theta}^t = \mathrm{ESTIMATE}(\Lambda_t, E, \bar{\theta}^{t-1})$

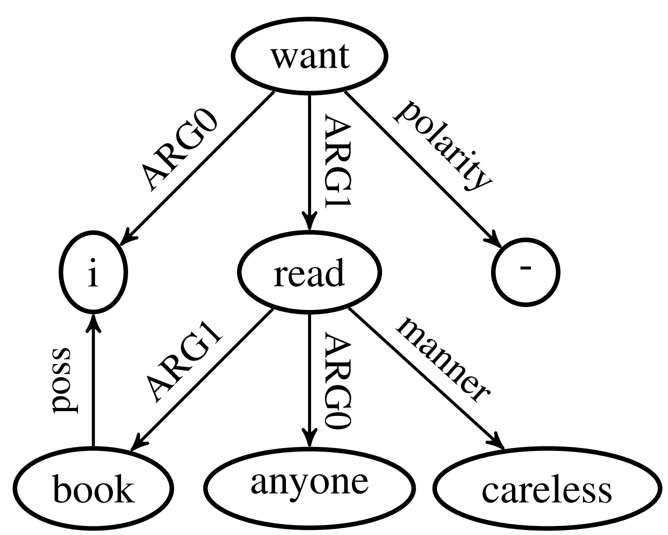
Evaluation results

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

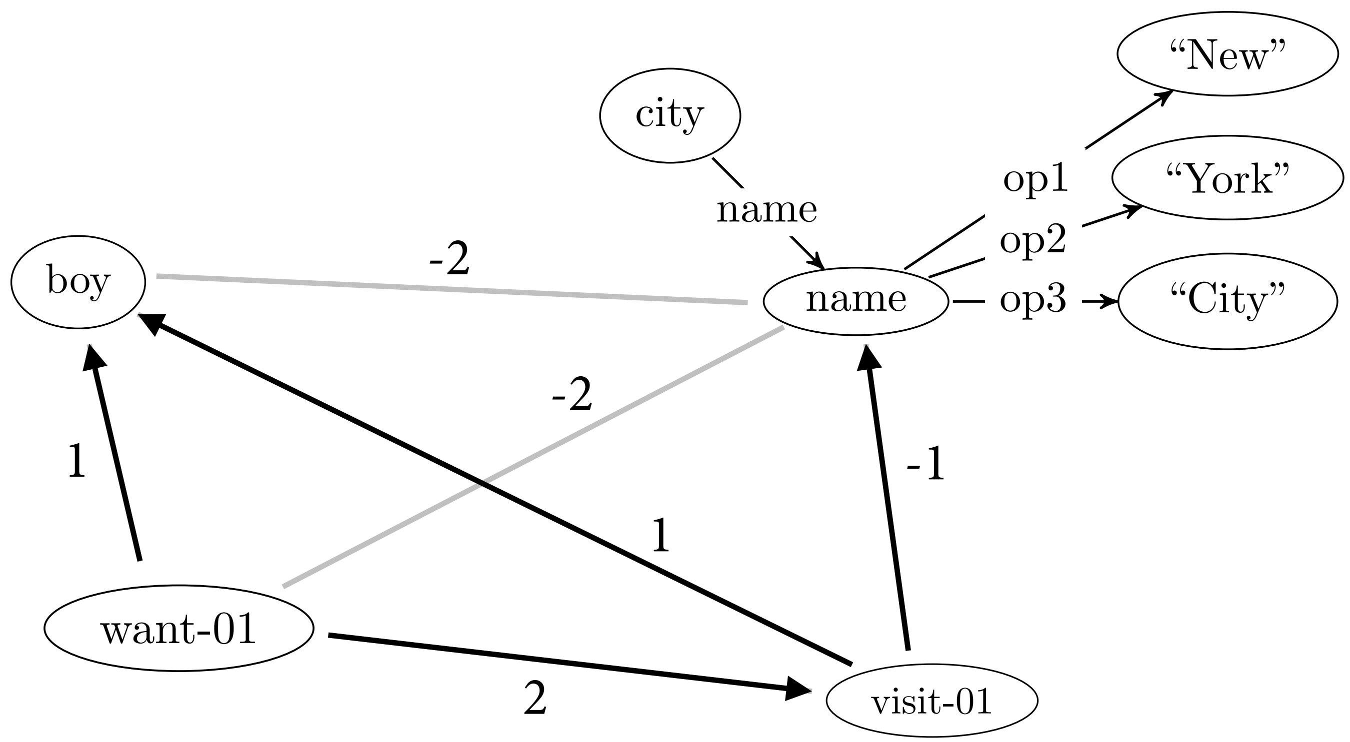
- Pros and cons of Geoquery:
  - semantic representations are trees — (too) simple
  - very small
- Since 2014, much larger corpora available: ~40k AMRs, graphs as semantic representations.



"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

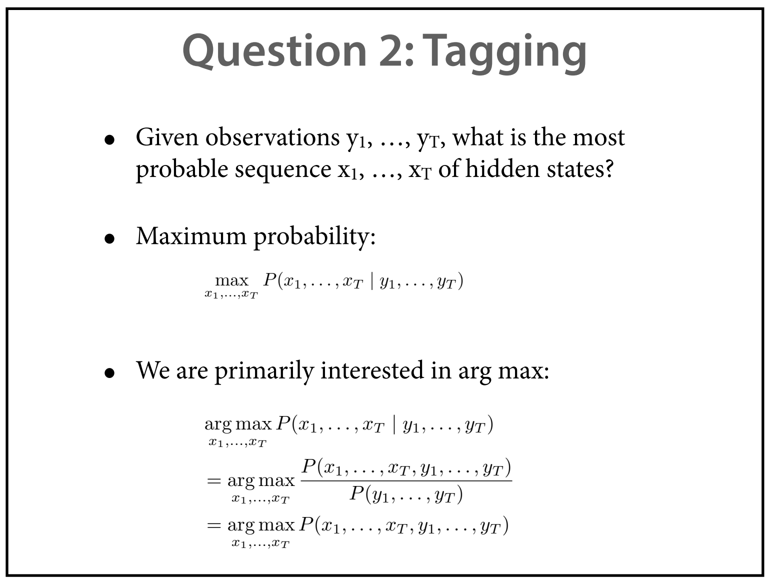
- JAMR can draw edge between any two nodes; syntactic structure of sentence used only indirectly.
- Semantic representations for words don't know anything about their semantic arguments.
- Edges for control verbs added arbitrarily, not because linguistic principle of control discovered.
- No notion of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

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.

slide credits

slides that look like this



come from

earlier editions of this class (ANLP), given by Alexander Koller

and their use is gratefully acknowledged. I try to make any modifications obvious, but if there are errors on a slide, assume that I added them.