Neural MRS parsing: Error analysis

Nick Chen

Jan Buys

Emily M. Bender

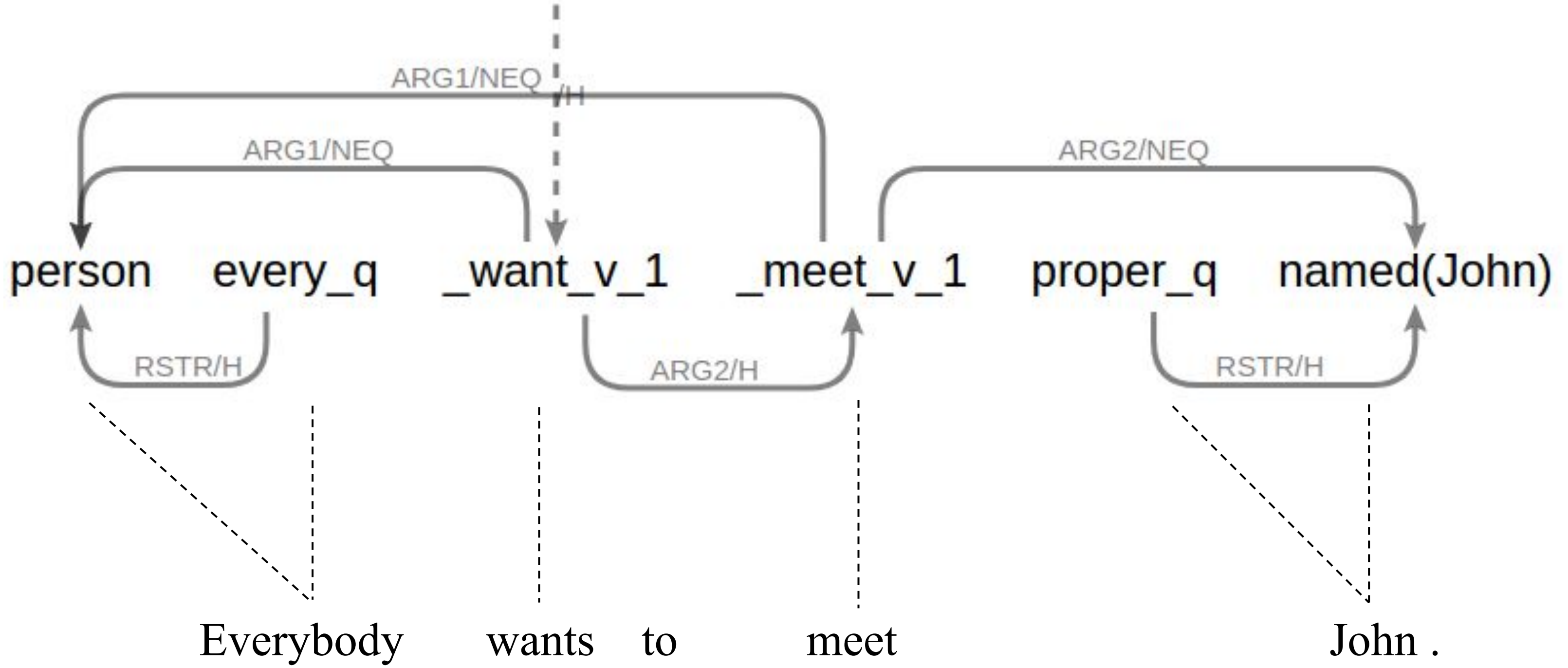
PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Deep Deep parsing
(Buys and Blunsom, ACL 2017)

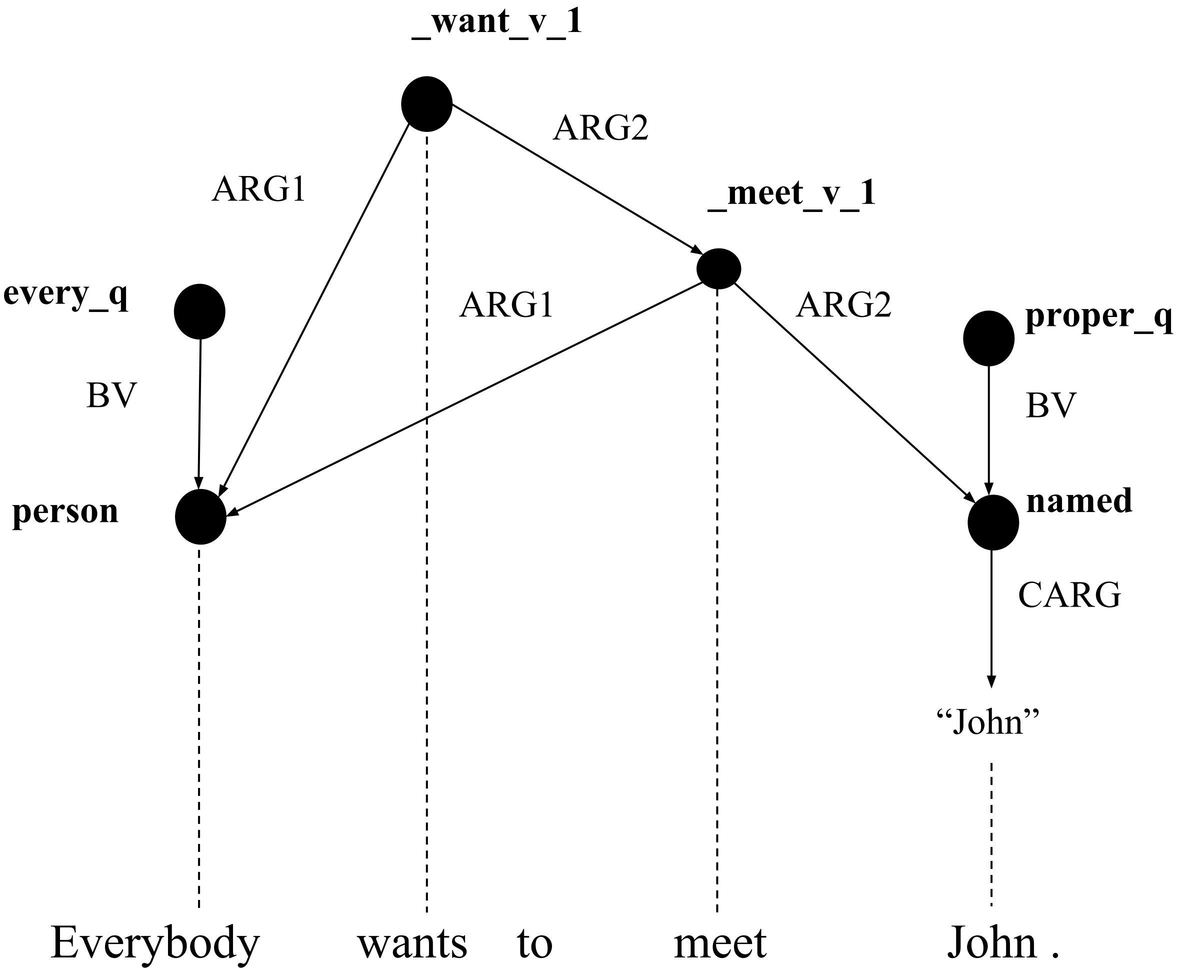
Fast, accurate, robust

- Deep learning models to transform natural language sentences into graph-based meaning representations
- Transition-based neural parsers applicable to multiple graph-based semantic formalisms
- Encoder-decoder RNNs with hard attention, pointer networks and a stack-based architecture

Dependency MRS (DMRS)



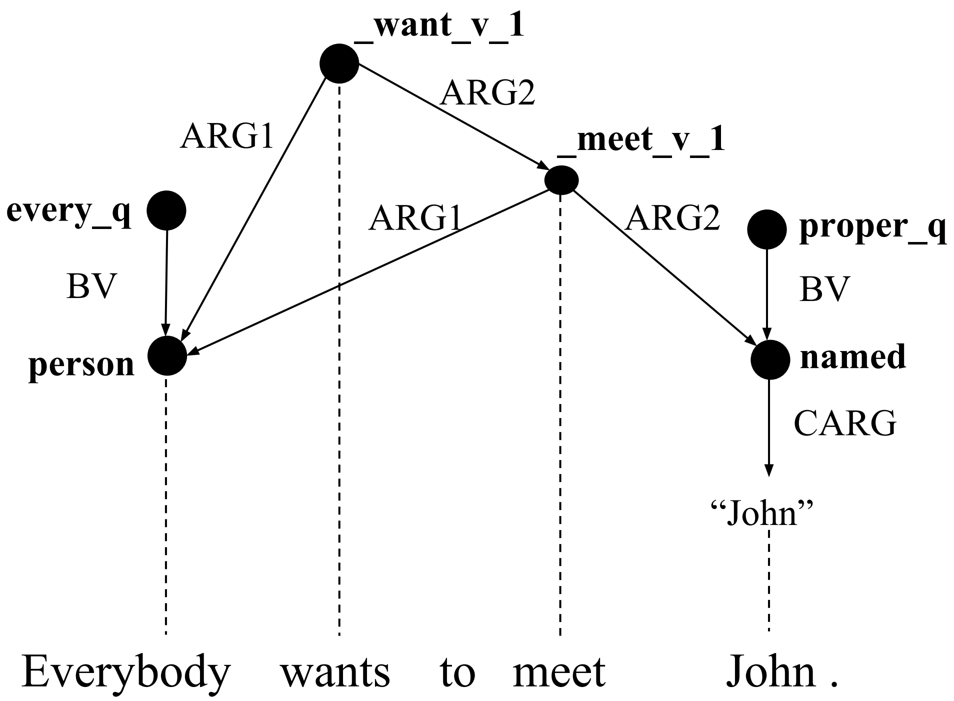
Semantic Graphs



End-to-end semantic graph parsing

Top-down graph linearization

```
:root( <2> _want_v_1
  :ARG1( <1> person
    :BV-of( <1> every_q ) )
  :ARG2 <4> _meet_v_1
    :ARG1*( <1> person
    :ARG2( <5> named_CARG
      :BV-of ( <5> proper_q ) ) )
```



Transition-based graph parsing

- Arc-eager transition system for semantic graphs
- Data structures: Input sentence, stack, buffer
- Actions:
  - Shift - generate next predicate on buffer
  - Reduce
  - Left-arc
  - Right-arc
  - Cross-arc

Transition-based parsing

Everybody wants to meet John .

Transition	Stack	Buffer
Init(1, person)		(1, person)

Transition-based parsing

Everybody wants to meet John .

Transition	Stack	Buffer
Shift(1, every_q)	(1, person)	(1, every_q)

Transition-based parsing

Everybody wants to meet John .

Transition	Stack	Buffer
Left-arc(BV)	(1, person)	(1, every_q)

Transition-based parsing

Everybody **wants** to meet John .

Transition	Stack	Buffer
Shift(2, _v_1)	(1, person), (1, every_q)	(2, _want_v_1)

Transition-based parsing

Everybody **wants** to meet John .

Transition	Stack	Buffer
Reduce	(1, person)	(2, _want_v_1)

Transition-based parsing

Everybody **wants** to meet John .

Transition	Stack	Buffer
Left-arc(ARG1)	(1, person)	(2, _want_v_1)

Transition-based graph parsing

Transition-based parsing: Oracle

- Node ordering - monotone ordering wrt alignments
- Predict alignment spans start (shift) and end (reduce)

Delexicalization: Lemmas are predicted separately and recovered during post-processing

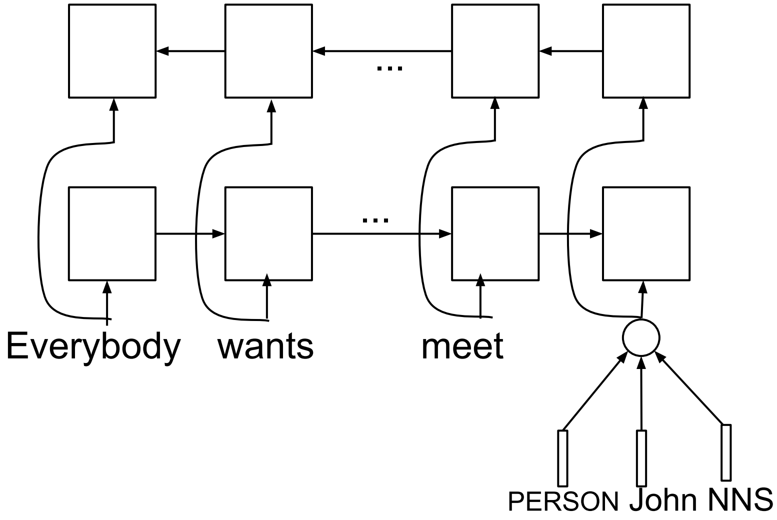
End-to-end graph parsing: Encoder-decoders

Formulate parsing as a sequence to sequence problem

- LSTM Recurrent neural network encodes the sentence
- A decoder LSTM predicts graph linearization
- Attention mechanism links the encoder and decoder

End-to-end graph parsing

Bidirectional RNN encoder



Graph parsing with stack-based decoders

- Decoder LSTM predicts actions and predicates
- Model the transition system stack and buffer
- Use the alignments of top stack and buffer nodes to extract encoder features

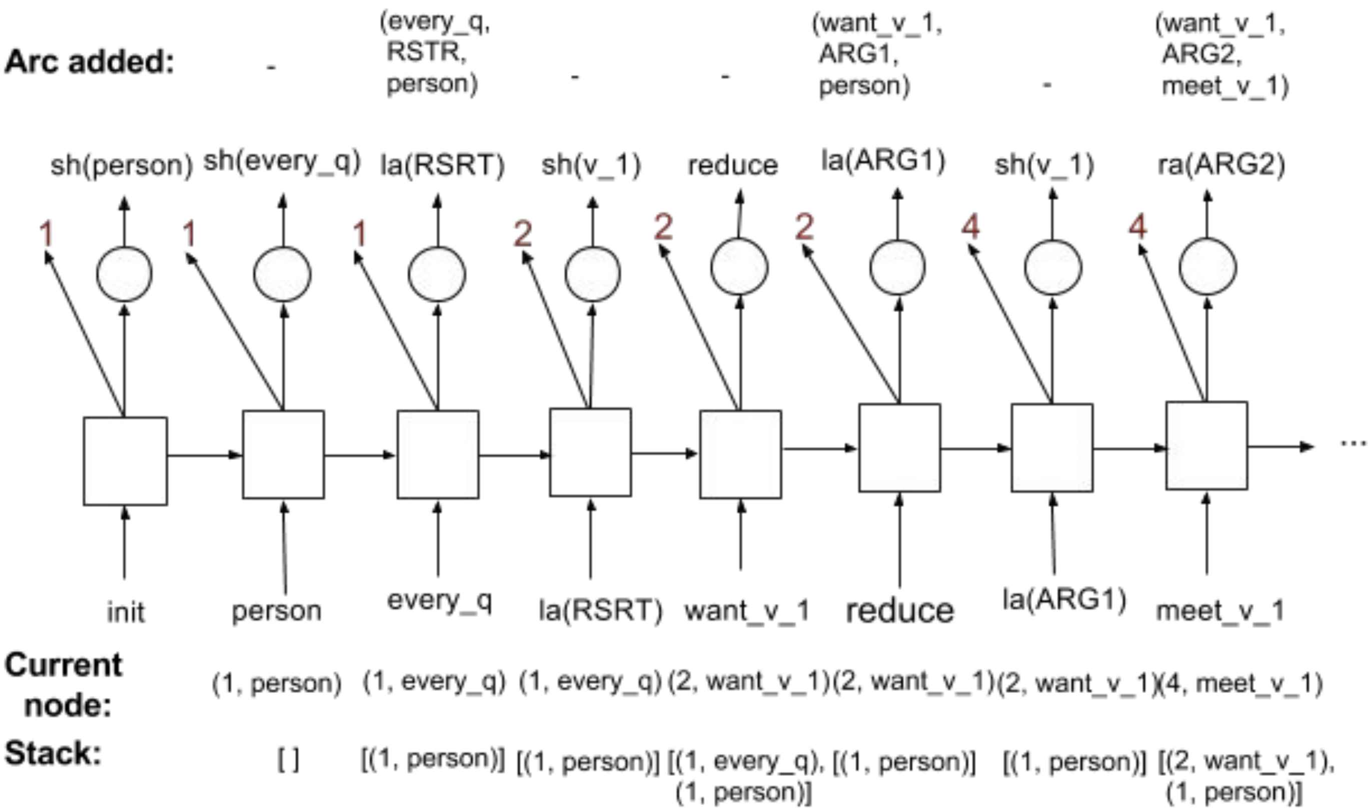
Graph parsing with stack-based decoders

RNN decoder with hard attention

Input sentence **e**, transition sequence **t**, alignment **a**.

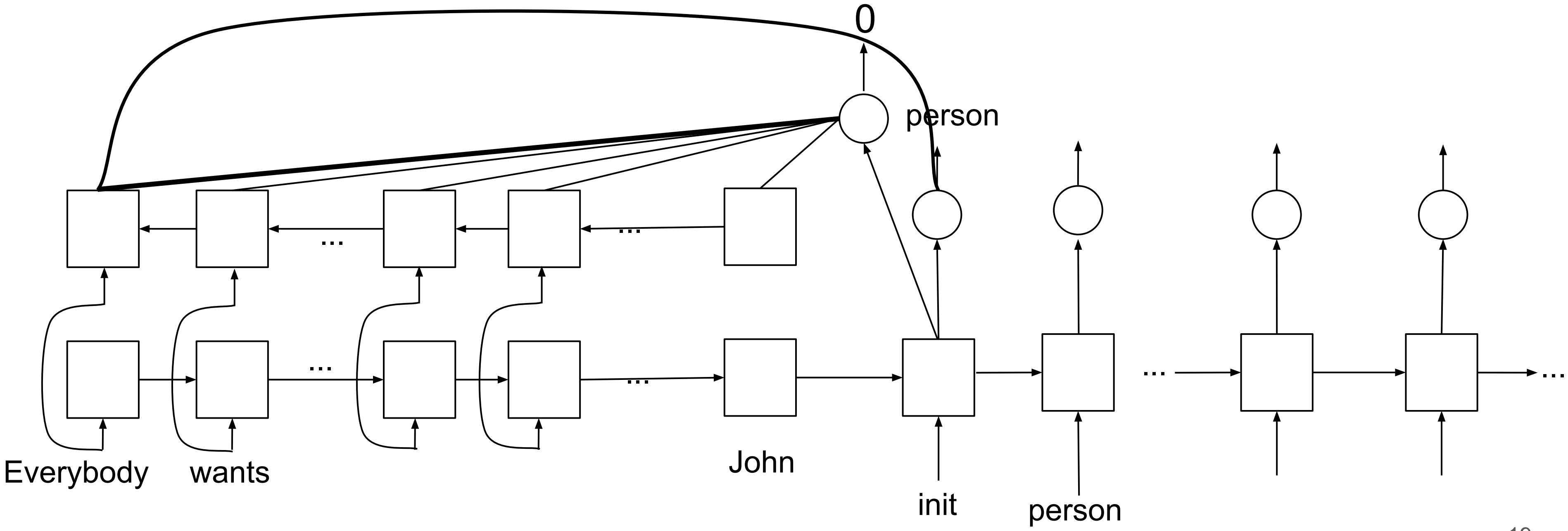
$$p(\mathbf{t}, \mathbf{a}|\mathbf{e}) = \prod_{j=1}^{J} p(a_j|(\mathbf{a}, \mathbf{t})_{1:j-1}, \mathbf{e}) p(t_j|\mathbf{a}_{1:j}, \mathbf{t}_{1:j-1}, \mathbf{e}).$$

Graph parsing with stack-based decoders



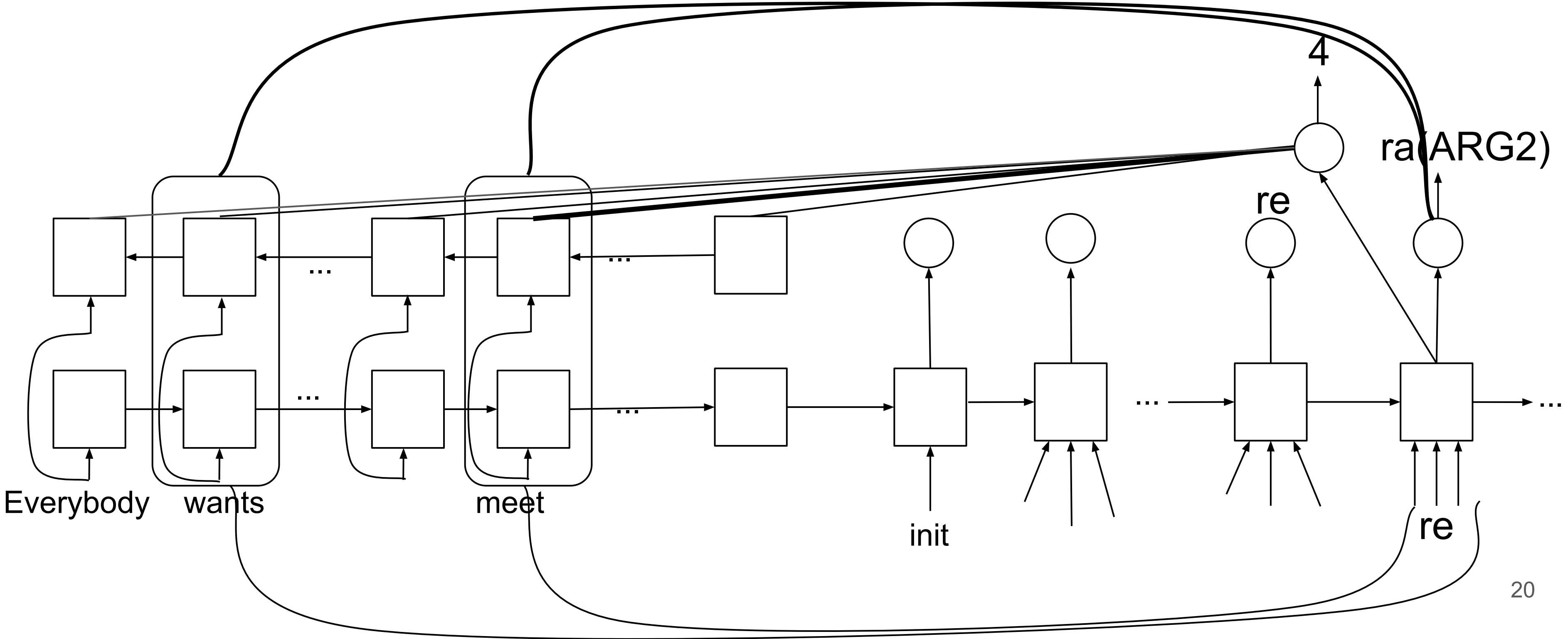
Graph parsing with stack-based encoder-decoders

RNN decoder with hard attention



Graph parsing with stack-based encoder-decoders

RNN decoder with stack-based features



DMRS Experiments

Encoder-decoders with pointer networks for alignment

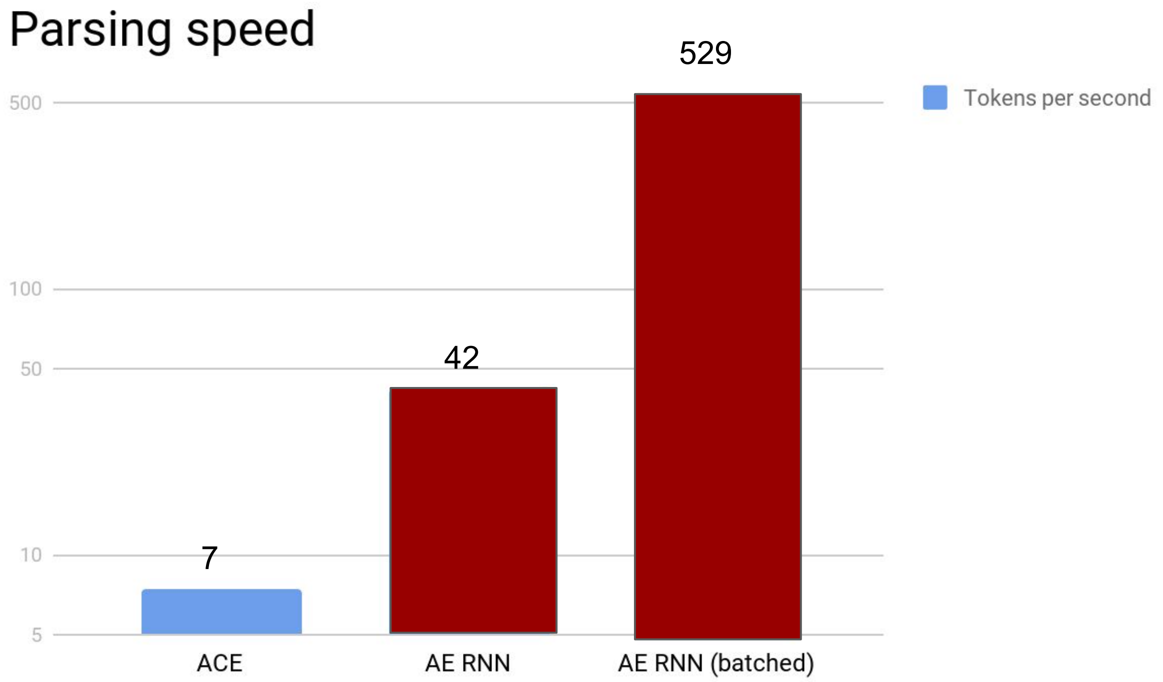
Model	**EDM**	**EDM Predicates**	**EDM Arguments**
Top-down soft att	81.53	85.32	76.94
Arc-eager soft att, lexicalized	81.35	85.79	76.02
Arc-eager soft att, unlexicalized	82.56	86.76	77.54
Arc-eager hard att	84.65	87.77	80.85
Arc-eager stack-based att	**85.28**	**88.38**	**81.51**

DMRS Experiments

Test results

Model	**EDM**	**EDM Predicates**	**EDM Arguments**	**Smatch**
Top-down RNN	79.68	83.36	75.16	85.28
Arc-eager RNN	**84.16**	**87.54**	**80.10**	**86.69**
ACE (ERG)	89.64	92.08	86.77	93.50

DMRS Experiments



Why error analysis?

Does the Deep Deep parser generate ERSs that are ill-formed or otherwise would never be produced by the grammar?

Or is it merely a matter of ‘ordinary’ attachment errors...

If so, can we extract wellformedness conditions that are violated and then use these to inform the Deep Deep parser?

Error analysis: Methodology

Data: DeepBank; pre-defined dev set, ~1800 items

Compare EDM triples

Look for:

Mismatches of predicates

Mismatches of predicate-ARG-predicate triples

Overall numbers: Predicate symbols

Total	60710
Common	47473
Gold Only	6840
Surface	(37.11%) 2478
Abstract	(62.89%) 4200
System Only	6397
Surface	(38.14%) 2262
Abstract	(61.86%) 3669
Total	5931

Overall numbers: Incorrect ARG

ARG1	688
ARG2	245
ARG3	15
ARG	8
Total	956

Overall numbers: Extra ARG

ARG1	140
ARG2	57
ARG3	14
RSTR	37
Total	148

Predicate names

Error type: The Deep Deep parser uses lemmas to generate surface predicate names, which sometimes gives oddball results: _is_v_id; _to+order+to_x

Candidate lemmas come from a lookup table extracted from the training data (surface form -> lemma), using the Stanford CoreNLP lemmatizer as fallback.

Predicates senses are predicted without constraints, so no guarantee that lemma+sense combination will occur in the SEM-I.

Spurious predicate names: Examples

_to+order+to_x (98), _a_p (16), _is_v_id (15), _could_v_moda (12), _circa_v_modal (10), _term_a_1 (8), _term_a_of (7), _accord+to_p (7), _chip_a_1(6), ...

Predicate spans

Span start and end indexes are predicted before and after the predicate names, respectively.

Error type: The Deep Deep parser is willing to posit grammar preds like ‘compound’ and ‘appos’ over spans of a single token

Abstract predicates where the predicted span is a subsequence or supersequence of the gold span is a common source of errors.

Predicate span examples

Mevacor,	named proper_q	_mevacor/nnp_u_unknown udef_q
company	_company_n_of	_company_n_of
higher	comp _high_a_1	comp
drug,	_drug_n_1	_drug_n_1 udef_q
Merck	named proper_q	named proper_q compound
sales	_sale_n_of	_sale_n_of

Predicate span examples

's	poss def_explicit_q	poss def_explicit_q
at		_at_p
"		appos
a	_a_q	_a_q

Rival Boston Herald columnist Howie Carr, who usually rails at Statehouse "hacks" and nepotism, argued that the new drawings were designed to hide Mr. Madden's "rapidly growing forehead" and the facial defects of "chinless" Dan Shaughnessy, a Globe sports columnist. (item 103)

Extra ARGs

Cases where the Deep Deep parser posits an ARGn for a predicate that is not found in the gold annotations

In that particular instance, or never for the predicate

Extra ARG examples

udef_q ARG1 (14)

implicit_conj ARG1 (5)

pronoun_q ARG2 (2)

named ARG2 (1)

subord ARG3 (1)

compound RSTR (9)

At least one such example seems to be associated with a non-connected graph

Missing ARGs

The Deep Deep parser fails to include an ARG for a predicate that is in the gold

Future work: Classify into required (would always be present) v. optional arguments

Future work: Analysis of parser actions that lead to this outcome

Incorrect ARGs

Cases where the Deep Deep parser uses a legitimate ARG label but gives it the wrong value --- these are likely to be simple attachment errors

Future work: Look at these more carefully to see if there are type constraints on the ARGs (e v. x v. h in the MRS) that are being violated

Next steps

Do incorrect/missing/extra ARG errors lead to follow-on attachment errors?

What are the highest value errors to try to correct?

Which kinds of linguistic clues can we hand to the Deep Deep parser?