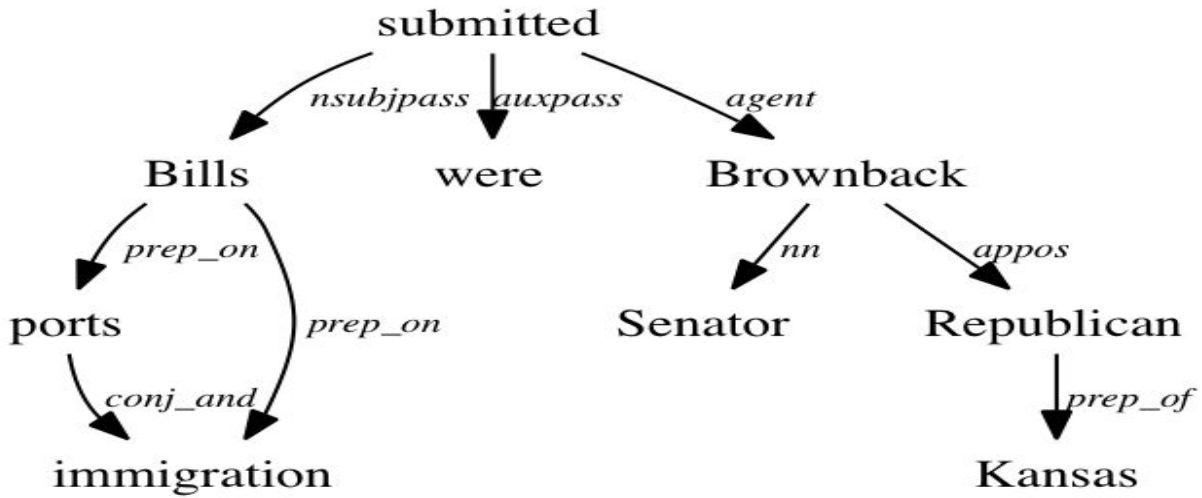
Robust Incremental Neural Semantic Graph Parsing

Jan Buys and Phil Blunsom

Dependency Parsing vs Semantic Parsing

- Dependency parsing models the syntactic structure between words in a sentence.

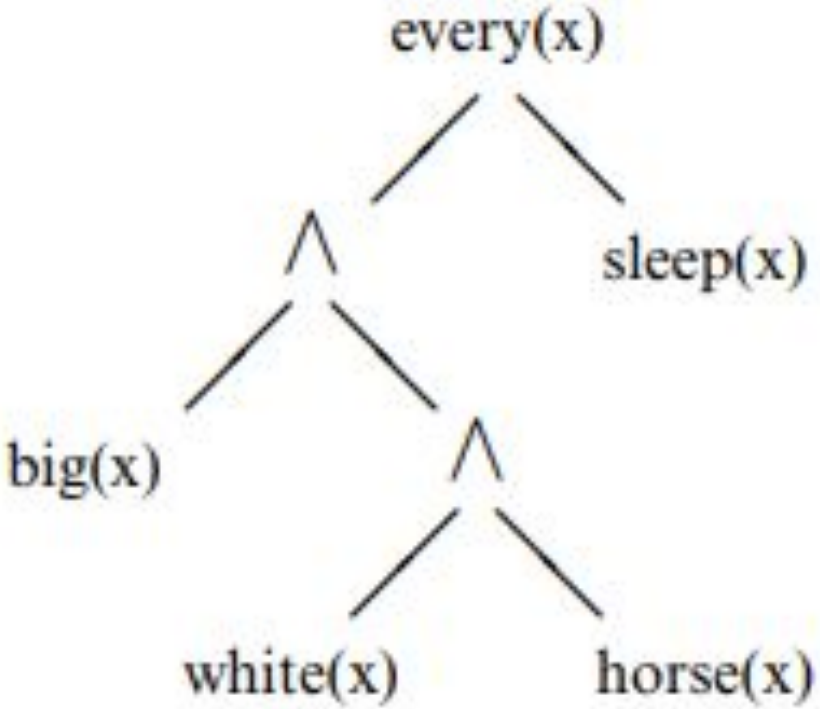


Dependency Parsing vs Semantic Parsing

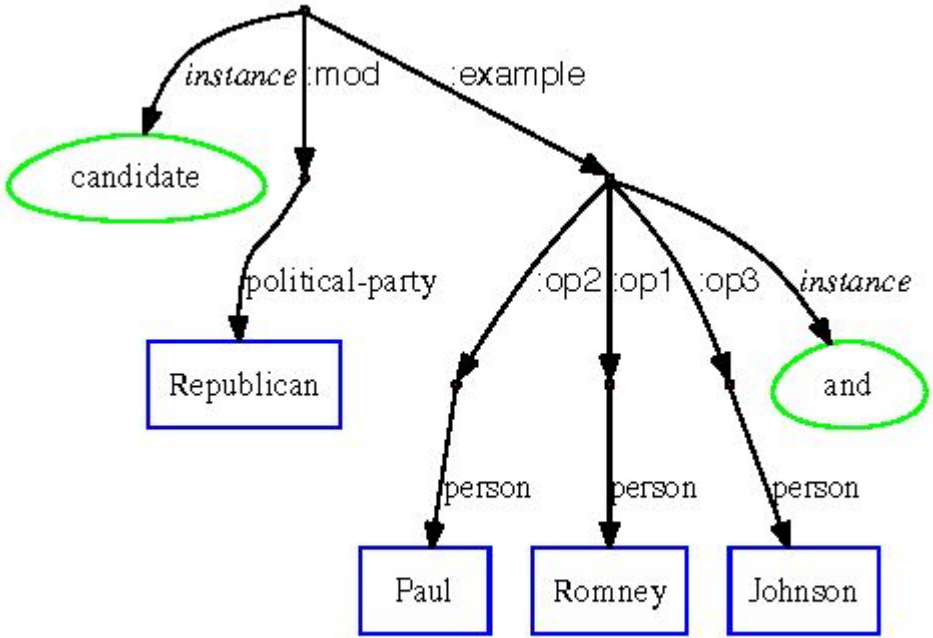
- Semantic parsing is converting sentences into structured semantic representations.

Semantic representations

- There are many ways to represent semantics.
- The author focuses on two types of semantic representations:
  - Minimal Recursion Semantics (MRS)
  - Abstract Meaning Representation (AMR)
- This paper uses two graph based conversions of MRS, Elementary Dependency Structure (EDS) and Dependency MRS (DMRS)

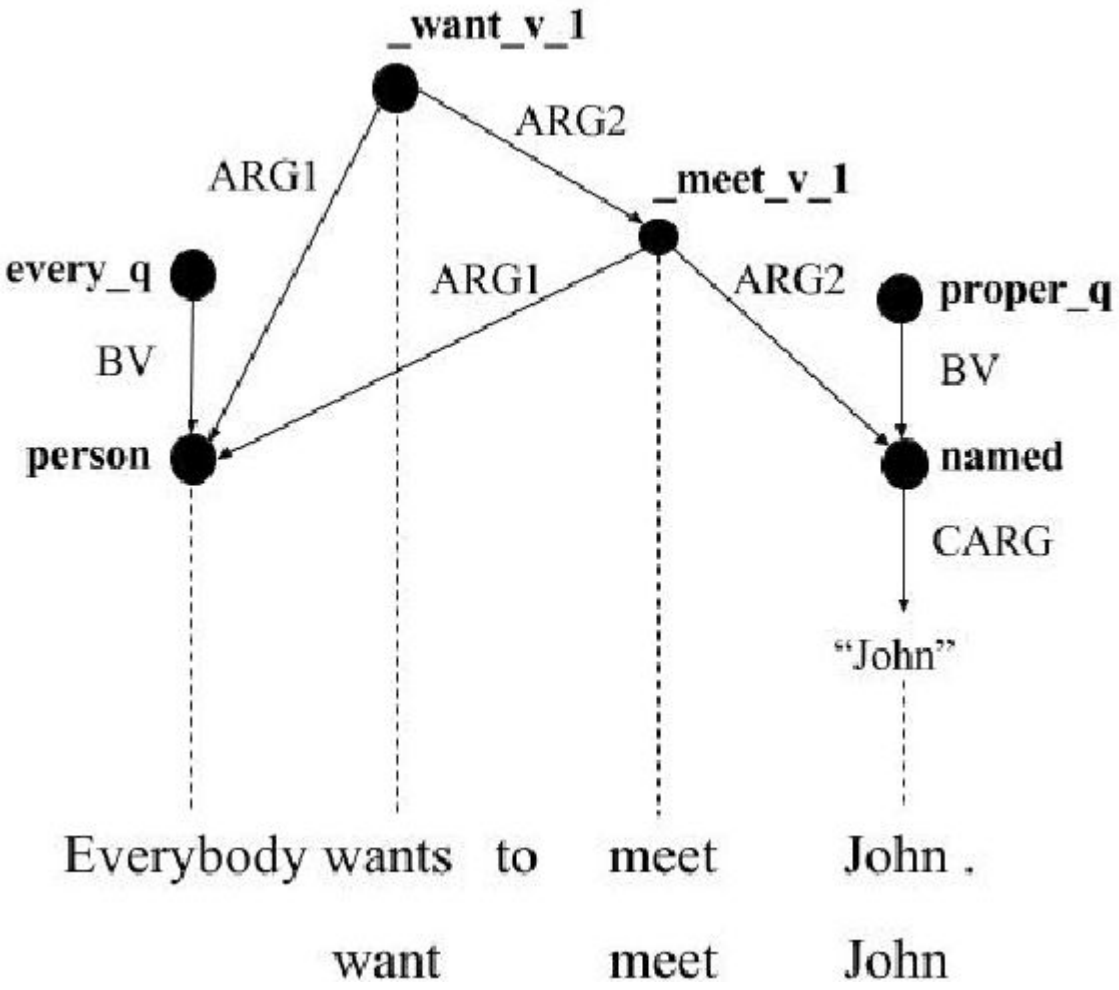


MRS



(d) *Republican candidates like* ***Romney***, ***Paul***, *and* ***Johnson***...

AMR



MRS+AMR

This graph is based on EDS and can be understood as AMR.
Node labels are referred to as predicates (concepts in AMR) and edge labels as arguments (AMR relations).

Model

- Goal:
  - Capture graph structure
  - Aligning words with vertices
  - Model linguistically deep representations

Incremental Graph Parsing

- Parse sentences to meaning representations by incrementally predicting semantic graphs together with their alignment.

Incremental Graph Parsing

Let $\mathbf{e} = \mathrm{e}_1, e_2, ..., e_I$ be a tokenized English sentence,

$\mathbf{t} = \mathrm{t}_1, t_2, ..., t_J$ its sequential representation of its graph derivation and $\mathbf{a} = \mathrm{a}_1, a_2, ..., a_J$ its alignment, then the conditional distribution is modeled as

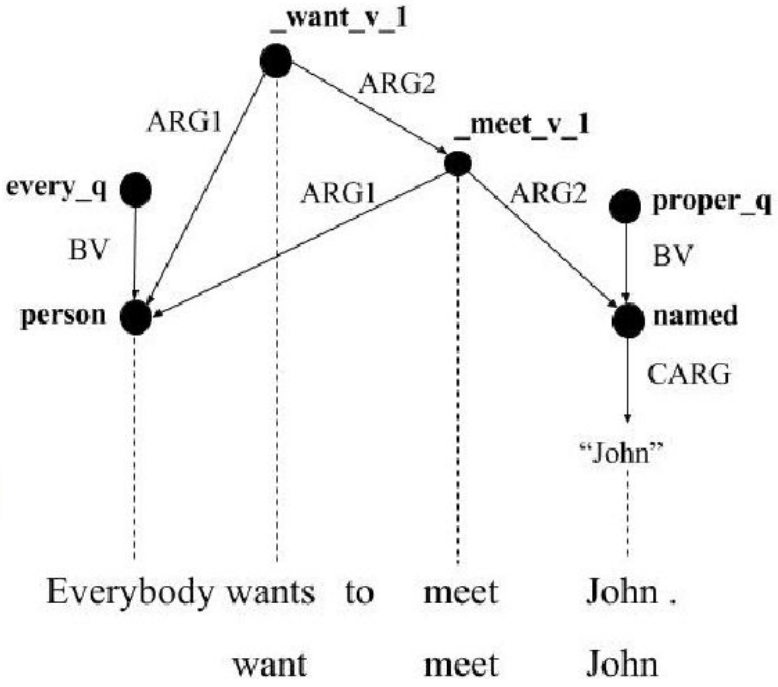
$$\mathrm{p}(\mathbf{t}, \mathbf{a} | \mathbf{e}) = \prod_{j=1}^{J} p(a_j \mid (\mathbf{a}, \mathbf{t})_{1:j-1}, \mathbf{e}) * p(t_j \mid \mathbf{a}_{1:j}, \mathbf{t}_{1:j-1}, \mathbf{e})$$

I is the number of tokens in a sentence
J is the number of vertices in the graph

Graph linearization(Top down linearization)

- Linearize a graph as the preorder traversal of its spanning tree

```
:root( <2> _v_1
  :ARG1( <1> person
    :BV-of( <1> every_q ) )
  :ARG2 <4> _v_1
    :ARG1*( <1> person
    :ARG2( <5> named_CARG
      :BV-of ( <5> proper_q ) )
```



Transition based parsing(Arc-eager)

- Interpret semantic graphs as dependency graphs.
- Transition-based parsing has been used extensively to predict dependency graphs incrementally.
- Arc-eager transition system on graphs.
- Condition on the generation of the sentence, generate nodes incrementally.

Stack, buffer, arcs

Action	Stack	Buffer	Arc added
init(1, person)	[ ]	(1, 1, person)	-
sh(1, every_q)	[(1, 1, person)]	(2, 1, every_q)	-
la(BV)	[(1, 1, person)]	(2, 1, every_q)	(2, BV, 1)
sh(2, _v_1)	[(1, 1, person), (2, 1, every_q)]	(2, 1, _v_1)	-
re	[(1, 1, person)]	(3, 2, _v_1)	-
la(ARG1)	[(1, 1, person)]	(3, 2, _v_1)	(3, ARG1, 1)

Transition actions: Shift, Reduce, Left Arc, Right Arc

Root, Cross Arc

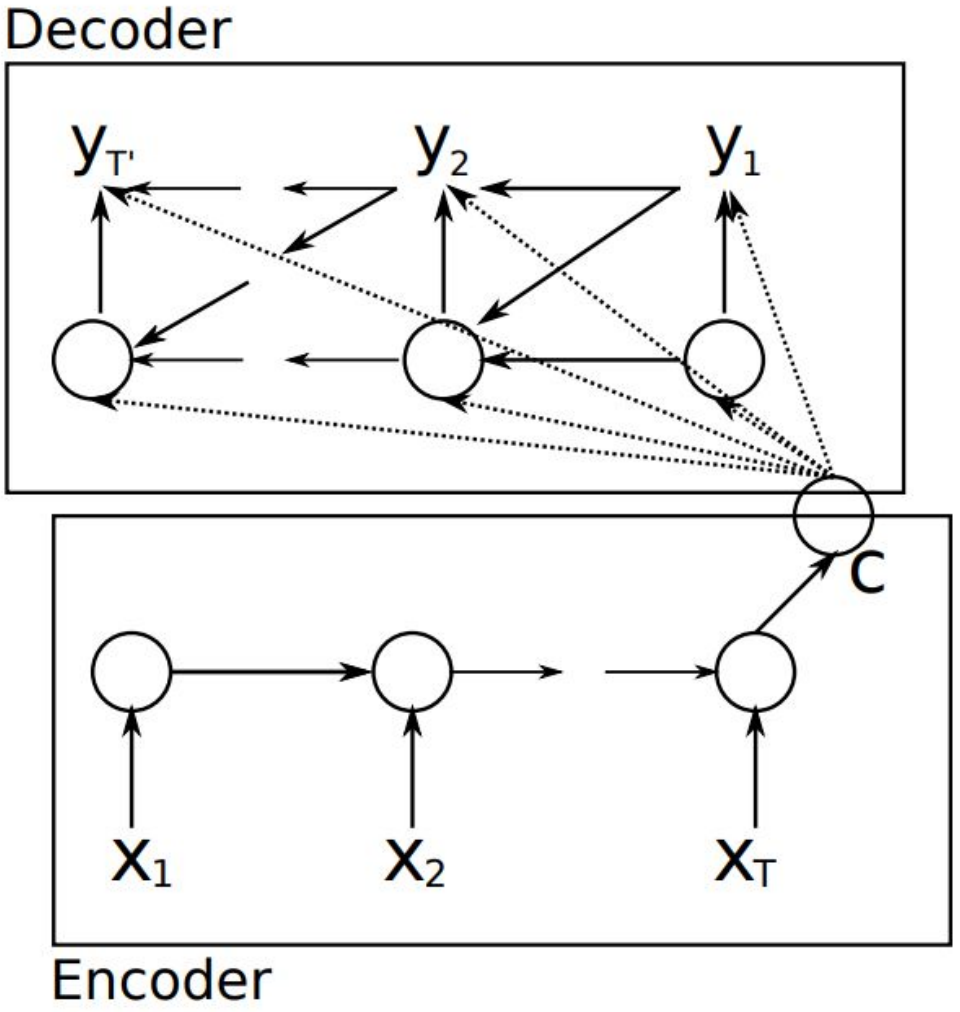
Model

- Goal:
  - Capture graph structure
  - Aligning words with vertices
  - **Model linguistically deep representations**

RNN-Encoder-Decoder

- Use RNN to capture deep representations.
- LSTM without peephole connections
- For every token in a sentence, embed it with its word vector, named entity tag and part-of-speech tag.
- Apply a linear transformation to the embedding and pass to a Bi-LSTM.

RNN-Encoder-Decoder



RNN-Encoder-Decoder

- Hard attention decoder with a pointer network.
- Use encoder and decoder hidden states to predict alignments and transitions.

Stack-based model

- Use the corresponding embedding of the words that are aligned with the node on top of the stack and the node in the buffer as extra features.
- The model can still be updated via mini batching, making it efficient

Data

- DeepBank (Flickinger et al., 2012) is an HPSG and MRS annotation of the Penn Treebank Wall Street Journal (WSJ) corpus.
- For AMR parsing we use LDC2015E86, the dataset released for the SemEval 2016 AMR parsing Shared Task (May, 2016).

Evaluation

- Use Elementary Dependency Matching (EDM) for MRS-based graphs.
- Smatch metric for evaluating AMR graphs.

Model setup

- Grid search to find the best setup.
- Adam, lr=0.01, bs=54.
- Gradient clipping 5.0
- Single-layer LSTMs with dropout of 0.3
- Encoder and decoder embeddings of size 256
- For DMRS and EDS graphs the hidden units size is set to 256, for AMR it is 128.

Comparison of linearizations(DMRS)

- Standard attention based encoder-decoder. (Alignments are encoded as tokens in the linearizations).

Model	EDM	EDM_P	EDM_A
TD lex	81.44	85.20	76.87
TD unlex	81.72	85.59	77.04
AE lex	81.35	85.79	76.02
AE unlex	82.56	86.76	77.54

EDM, EDM_P, EMD_A are metrics for EDM.

- The arc-eager unlexicalized representation gives the best performance, even though the model has to learn to model the transition system stack through the recurrent hidden states without any supervision of the transition semantics.
- The unlexicalized models are more accurate, mostly due to their ability to generalize to sparse or unseen predicates occurring in the lexicon.

Comparison between hard/soft attention(DMRS)

Model	EDM	EDM_P	EDM_A
TD soft	81.53	85.32	76.94
TD hard	82.75	86.37	78.37
AE hard	84.65	87.77	80.85
AE stack	85.28	88.38	81.51

Comparison to grammar-based parser(DMRS)

Model	TD RNN	AE RNN	ACE
EDM	79.68	84.16	89.64
EDM_P	83.36	87.54	92.08
EDM_A	75.16	80.10	86.77
Start EDM	84.44	87.81	91.91
Start EDM_A	80.93	85.61	89.28
Smatch	85.28	86.69	93.50

- ACE grammar parser has higher accuracy. (The underlying grammar is exactly the same)
- Model has higher accuracy on start-EDM(Only considering the start of the alignment to match). Implying that the model has more difficult in parsing the end of the sentence.
- The batch version of this model parses 529.42 tokens per second using a batch size of 128. The setting of ACE for which the author uses to report accuracies parses 7.47 tokens per second.

Comparison to grammar-based parser(EDS)

- EDS is slightly simpler than DMRS.
- The authors model improved on EDS, while ACE did not.
- They hypothesize that most of the extra information in DMRS can be obtained through the ERG, to which ACE has access but their model doesn't.

Model	AE RNN	ACE
EDM	85.48	89.58
EDM_P	88.14	91.82
EDM_A	82.20	86.92
Smatch	86.50	93.52

Comparisons on AMR parsing

Model	Concept F1	Smatch
TD no pointers	70.16	57.95
TD soft	71.25	59.39
TD soft unlex	72.62	59.88
AE hard unlex	76.83	59.83
AE stack unlex	77.93	61.21

State of the art on Concept F1 score: 83%

Comparisons on AMR parsing

- Outperforms baseline parser
- Doesn't perform as well as models that use extensive external resources(syntactic parsers, semantic role labellers)
- Outperforms sequence to sequence parsers, and a Synchronous Hyperedge Replacement Grammar model that uses comparable external resource.

Model	Smatch
Flanigan et al. (2014)	56
Wang et al. (2016)	66.54
Damonte et al. (2017)	64
Peng and Gildea (2016)	55
Peng et al. (2017)	52
Barzdins and Gosko (2016)	43.3
TD no pointers	56.56
AE stack delex	60.11

Conclusions

- In this paper we advance the state of parsing by employing deep learning techniques to parse sentence to linguistically expressive semantic representations that have not previously been parsed in an end-to-end fashion.
- We presented a robust, wide-coverage parser for MRS that is faster than existing parsers and amenable to batch processing.

References

Original paper

http://demo.ark.cs.cmu.edu/parse/about.html

https://nlp.stanford.edu/software/stanford-dependencies.shtml

https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/

Wikipedia