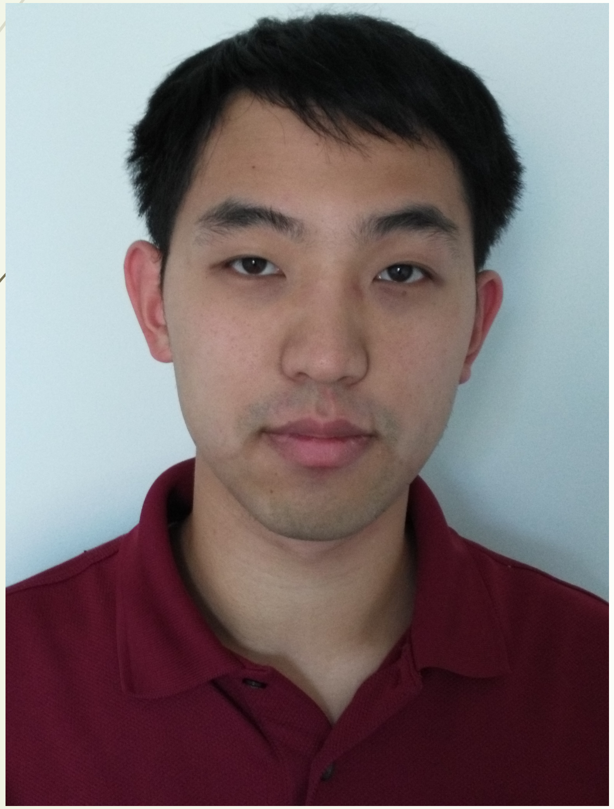


LSTM Hypertagging

Reid Fu, Ohio State Computer Science and Engineering

Michael White, Ohio State Linguistics and Facebook Conversational AI

INLG, 6 November 2018

LSTM Hypertagging Works!

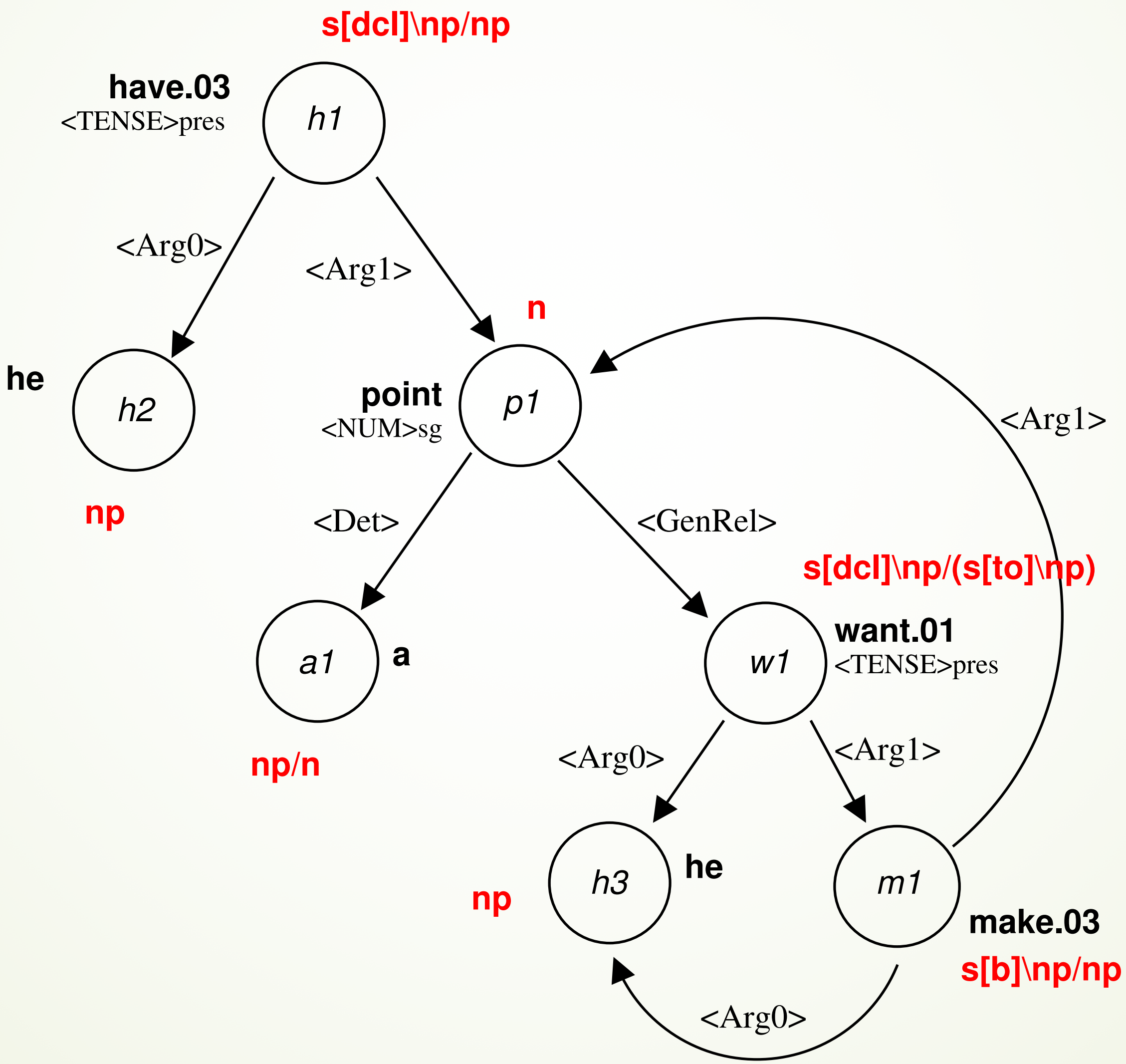
- Espinosa et al. (2008) coin the term ***hypertagging*** as short for supertagging for surface realization (aka fine-grained syntactic tagging a la Joshi and Bangalore)
- They show that maximum entropy hypertagging yields substantial performance improvements for **broad coverage** deep CCG realization
- More recently, Lewis et al. (2016) show large gains for CCG parsing using an **LSTM supertagger** instead of a maxent one
- We likewise show **large gains** in hypertagging accuracy and downstream realization quality with OpenCCG using an **LSTM hypertagger**
  - … especially when using **English-like input linearization**
  - ... yielding a 28% reduction in tagging error
  - … and an 8% increase in grammatically complete derivations
  - … leading to **substantially preferred** realizations

Who cares?

- Is anyone still doing grammar-based surface realization?

  (… stay tuned for discussion of related work at the end …)

The Task



s[dcl]\np/np
have.03
<TENSE>pres
h1
<Arg0>
<Arg1>
n
he
h2
np
point
<NUM>sg
p1
<Arg1>
<Det>
<GenRel>
s[dcl]\np/(s[to]\np)
a1
a
np/n
w1
want.01
<TENSE>pres
<Arg0>
<Arg1>
np
h3
he
m1
make.03
s[b]\np/np
<Arg0>

$$
\begin{array}{cccccccc}
\textit{He} & \textit{has} & \textit{a} & \textit{point} & \textit{he} & \textit{wants} & \textit{to} & \textit{make} \\
\mathsf{np} & \mathsf{s}_{dcl}\backslash\mathsf{np}/\mathsf{np} & \mathsf{np}/\mathsf{n} & \mathsf{n} & \mathsf{np} & \mathsf{s}_{dcl}\backslash\mathsf{np}/(\mathsf{s}_{to}\backslash\mathsf{np}) & \mathsf{s}_{to}\backslash\mathsf{np}/(\mathsf{s}_{b}\backslash\mathsf{np}) & \mathsf{s}_{b}\backslash\mathsf{np}/\mathsf{np} \\
 & & \multicolumn{2}{c}{\overline{\qquad\qquad}{>}} & \overline{\qquad}{>}\mathbf{T} & & \multicolumn{2}{c}{\overline{\qquad\qquad}{>}\mathbf{B}} \\
 & & \multicolumn{2}{c}{\mathsf{np}} & \mathsf{s}/(\mathsf{s}\backslash\mathsf{np}) & & \multicolumn{2}{c}{\mathsf{s}_{to}\backslash\mathsf{np}/\mathsf{np}} \\
 & & & & & \multicolumn{3}{c}{\overline{\qquad\qquad\qquad}{>}\mathbf{B}} \\
 & & & & & \multicolumn{3}{c}{\mathsf{s}_{dcl}\backslash\mathsf{np}/\mathsf{np}} \\
 & & & & \multicolumn{4}{c}{\overline{\qquad\qquad\qquad\qquad}{>}\mathbf{B}} \\
 & & & & \multicolumn{4}{c}{\mathsf{s}_{dcl}/\mathsf{np}} \\
 & & & & \multicolumn{4}{c}{\overline{\qquad\qquad\qquad\qquad}} \\
 & & & & \multicolumn{4}{c}{\mathsf{np}\backslash\mathsf{np}} \\
 & & \multicolumn{6}{c}{\overline{\qquad\qquad\qquad\qquad\qquad\qquad}{<}} \\
 & & \multicolumn{6}{c}{\mathsf{np}} \\
 & \multicolumn{7}{c}{\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad}{>}} \\
 & \multicolumn{7}{c}{\mathsf{s}_{dcl}\backslash\mathsf{np}} \\
\multicolumn{8}{c}{\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}{<}} \\
\multicolumn{8}{c}{\mathsf{s}_{dcl}}
\end{array}
$$

Predicted lexical categories are used in OpenCCG derivations

LSTM Hypertagger streamlines method

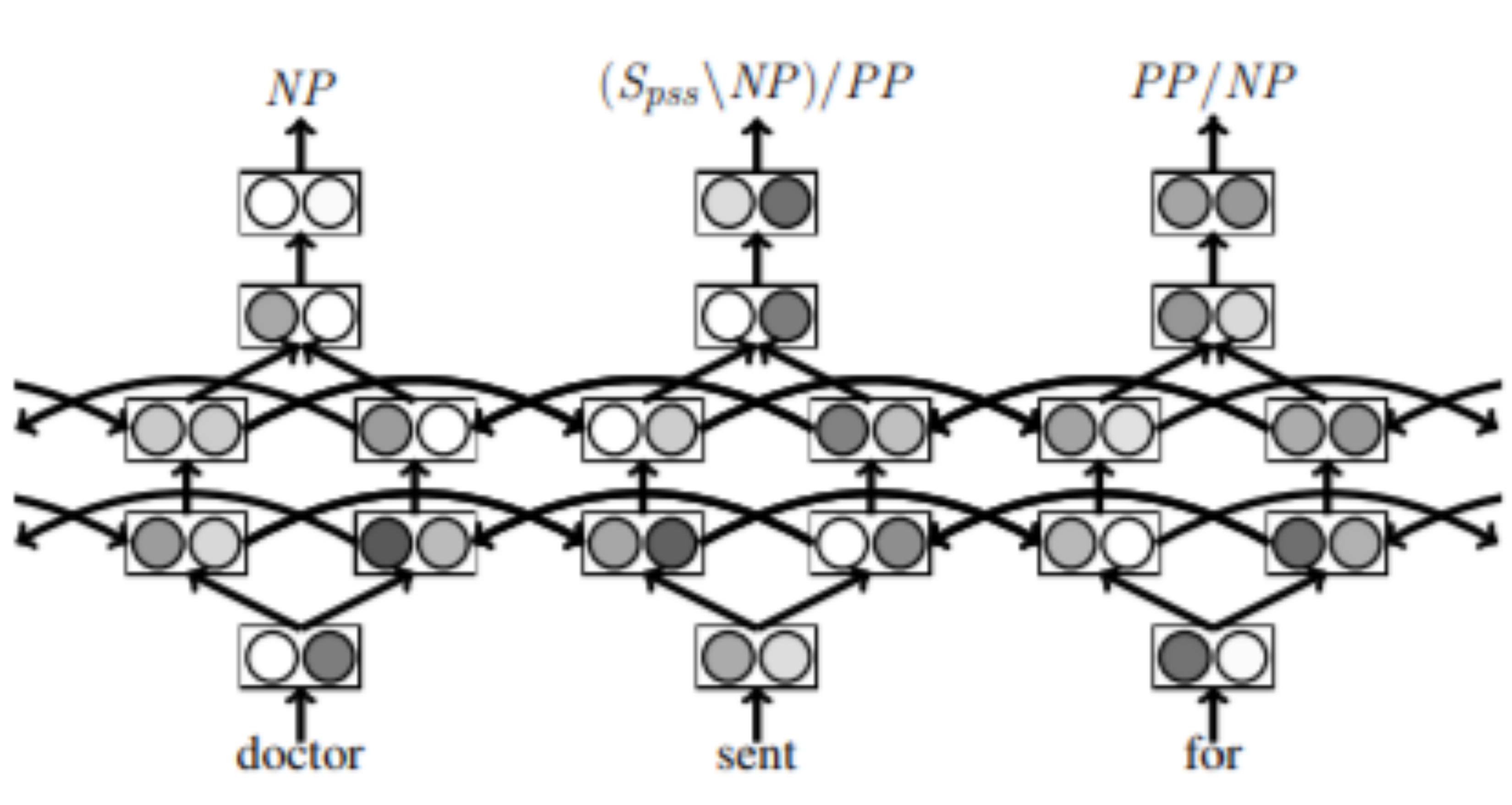
Maxent Hypertagger

- Uses graph-local features
- Original hypertagger first predicts POS tags, then uses graph-local POS tags to predict lexical categories (ie supertags)
- Unpublished two-stage hypertagger stacks on a second stage of predicting lex cats using initial graph-local supertags

LSTM Hypertagger

- Uses graph-local features
- Predicts lexical categories directly
- Derives contextual evidence via bi-LSTM

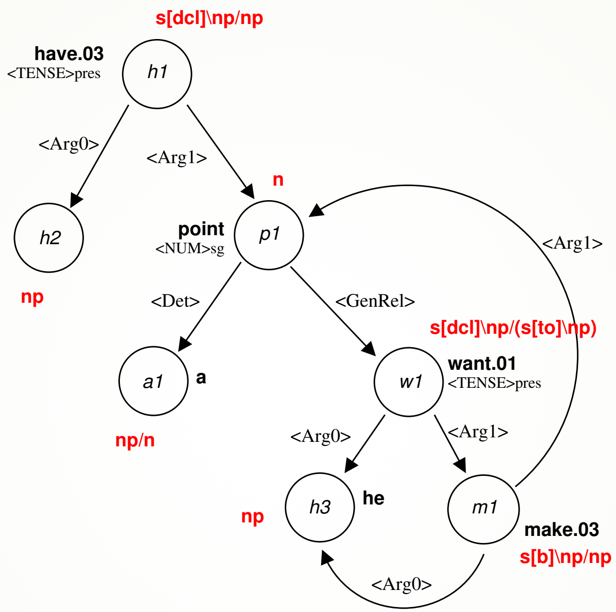
Lewis et al. Architecture (unchanged)



But our inputs are graphs?

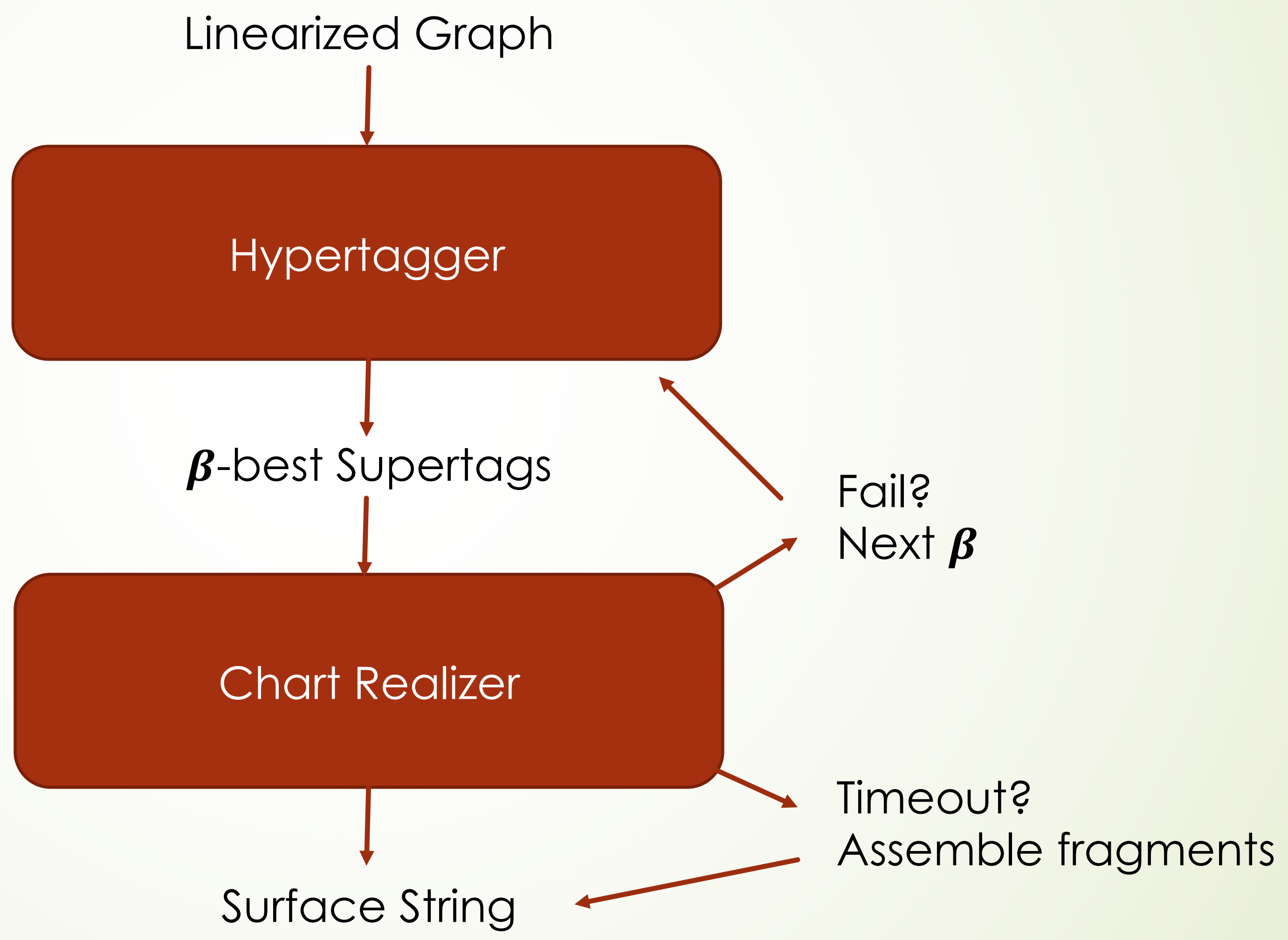
- Could try a **graph encoding** method (as in Marcheggiani & Perez-Beltrachini, INLG-18!)
- Or, could use **more conventional** bi-LSTM approach and leave a graph-based method for future work ☺
- Doing so requires the input graph to be **linearized**; we take inspiration from Konstas et al.'s (2017) AMR generation approach
- Unlike in their setting, here method of **ordering matters**:
  - Oracle ≫ English-like > depth-first ≫ random
- English-like ordering:
  - Det > Poss > Arg0 > Short Mods > Head > Arg1..5 > Short-to-long Mods

Input Linearization Example

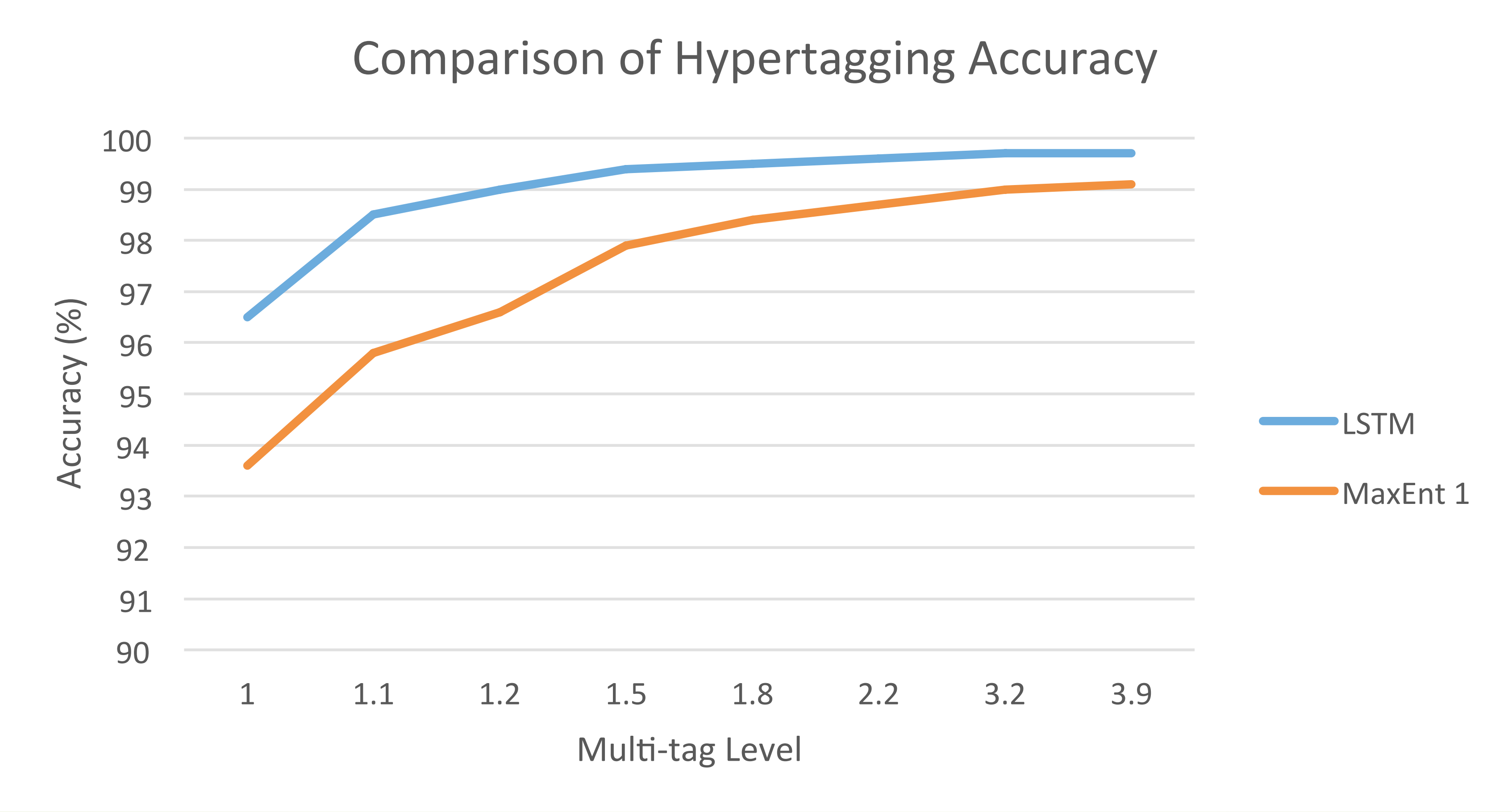


- (he[…] have[…] (a[…] point[…] he[…] want[…] make[…]))
  - where each node has graph-local features

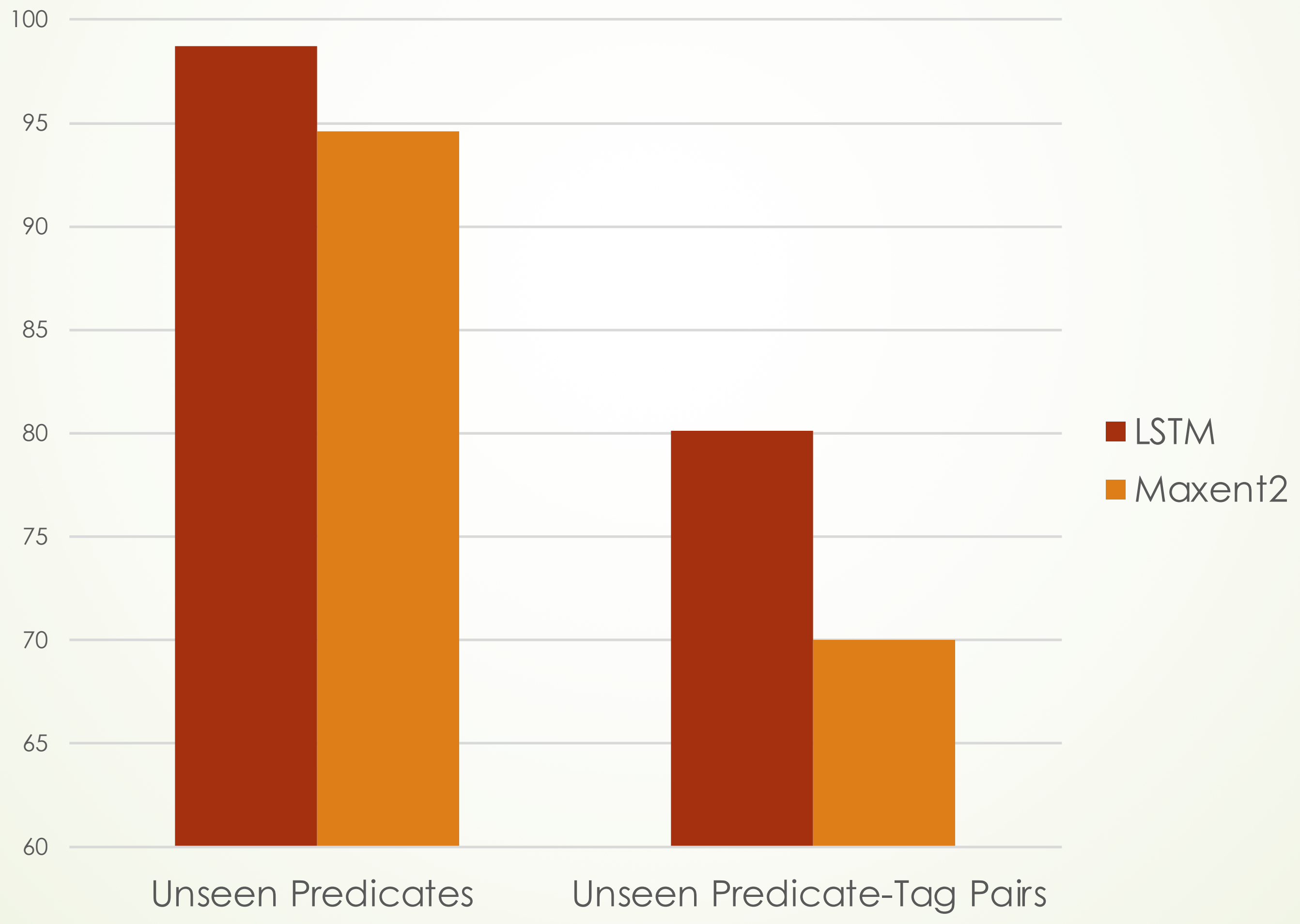
Experiments with OpenCCG CCGbank



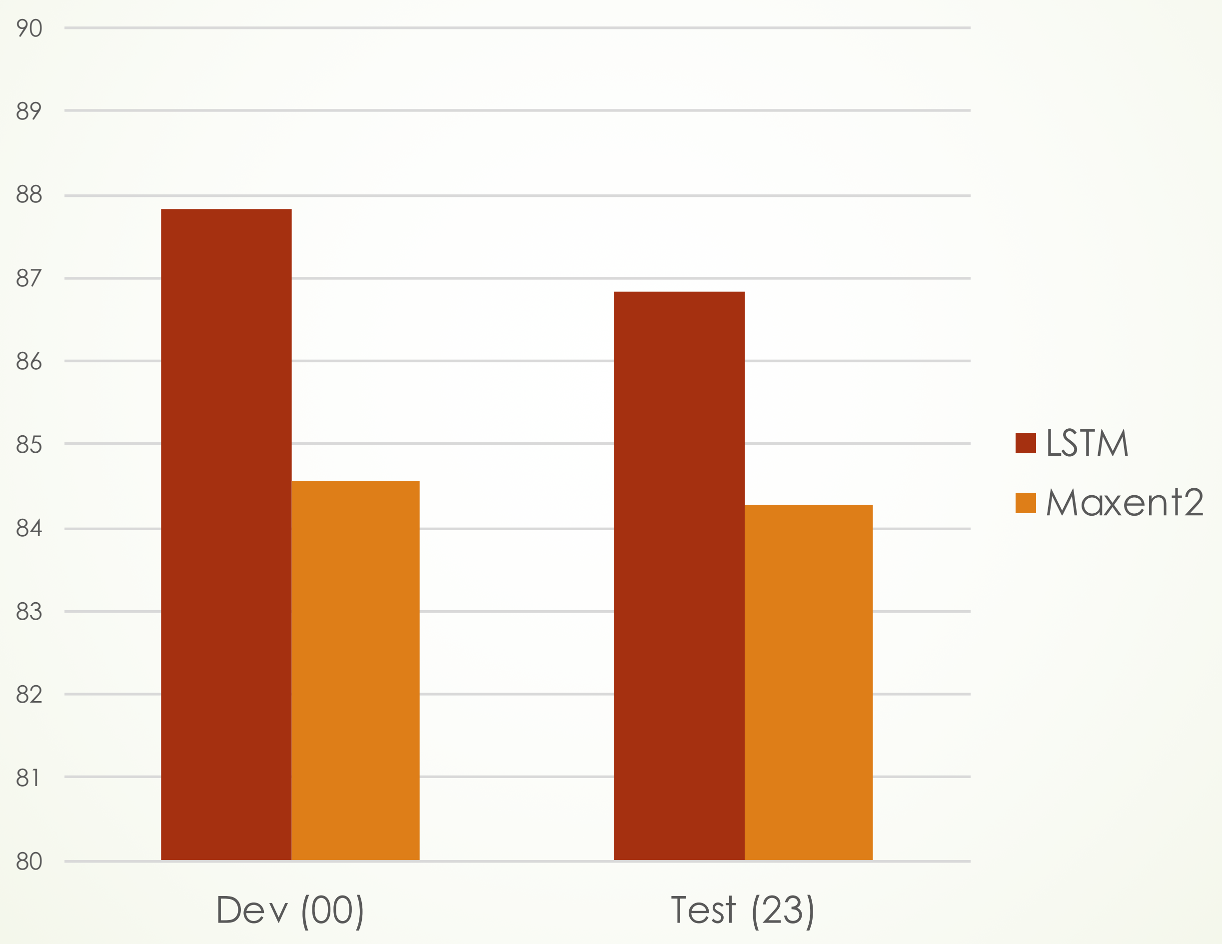
LSTM achieves high accuracy at much lower multitagging levels

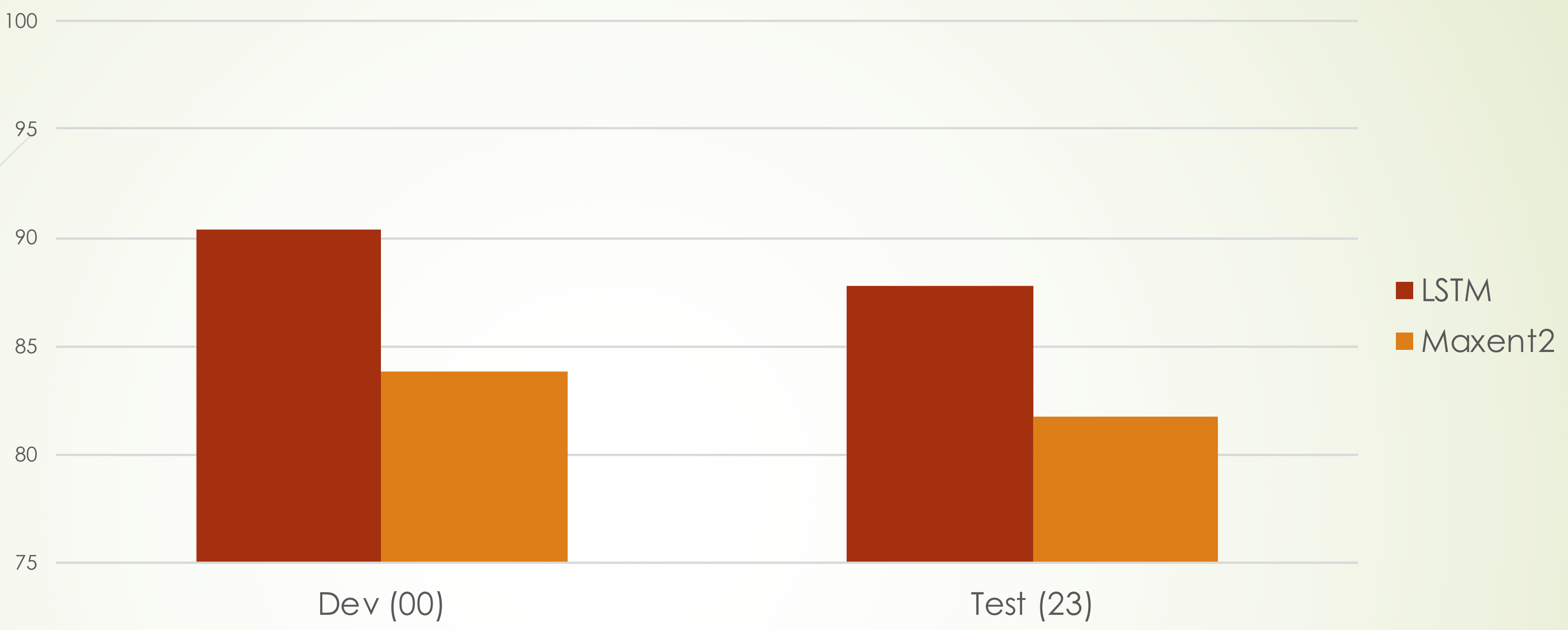


LSTM Hypertagger generalizes better to difficult cases



BLEU scores increase substantially (+2.5)





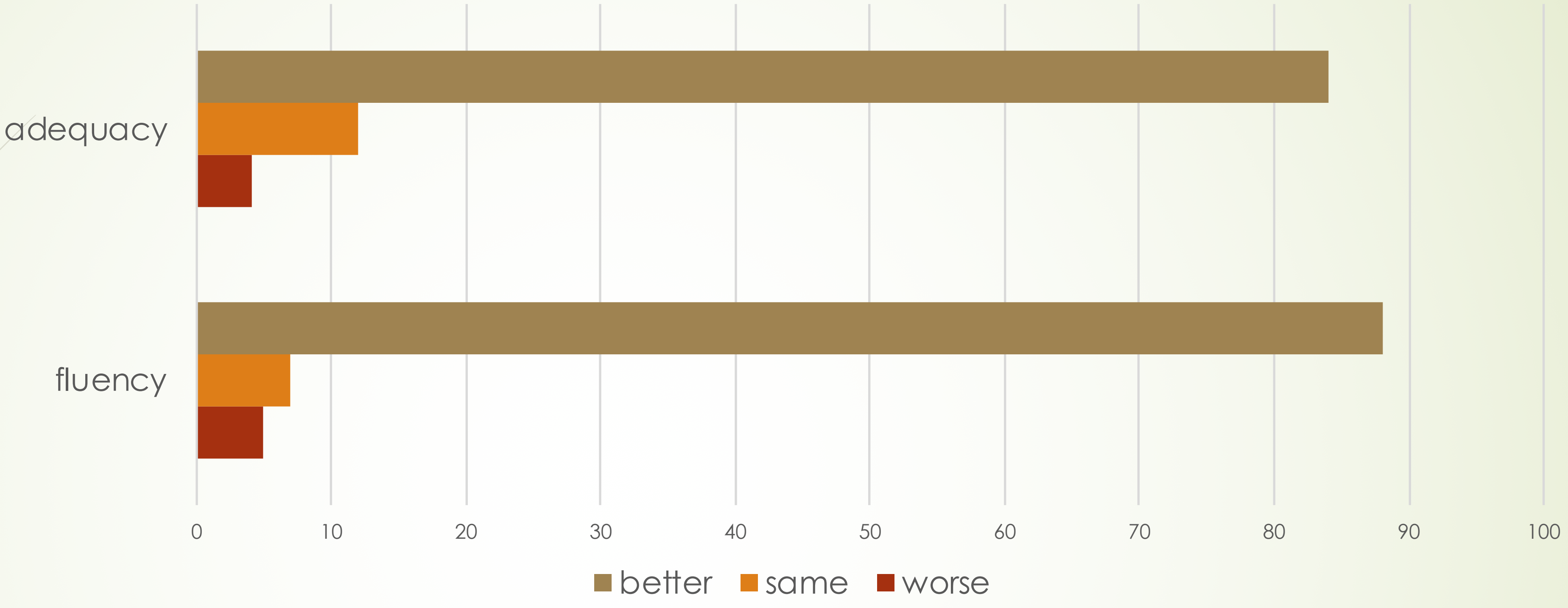
Many more complete derivations (+6)

An example that gets better

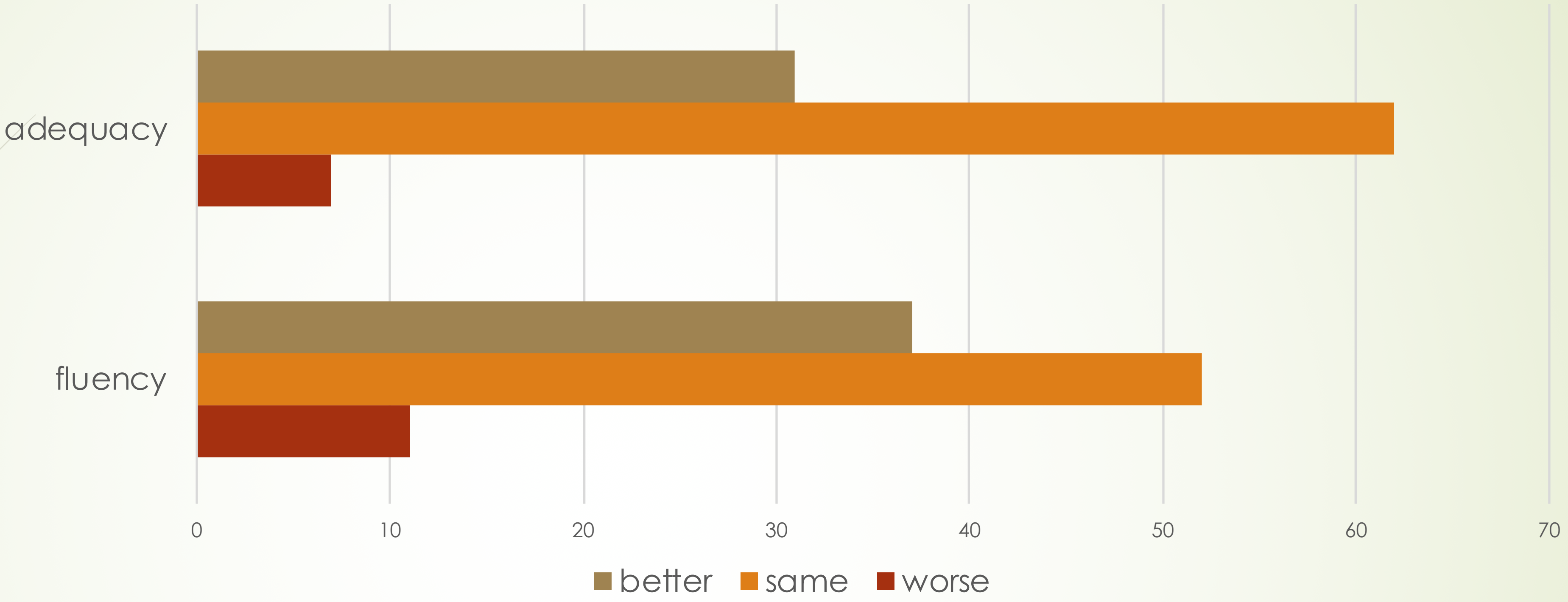
wsj 0004.8	nevertheless , said Brenda Malizia Negus , editor of Money Fund Report , yields may blip up again before they blip down because of recent rises in short-term interest rates .
LSTM	**yields nevertheless may** blip up again before they blip down because of recent rises in short-term interest rates , said Brenda Malizia Negus , editor of Money Fund Report .
Maxent2	**may nevertheless yields** , said Brenda Malizia Negus , editor of Money Fund Report , again blip up before they blip down because of recent rises in short-term interest rates .

Human evaluation focuses on change in complete derivations

- Two linguistics student judges, blind to the purpose of the study
- 100 randomly selected sentences in a random order where
  1. **Either** LSTM or Maxent2 yielded a **complete** derivation (but not both)
  2. **Both** LSTM and Maxent2 yielded a **complete** derivation **or neither** did
- Judges independently chose better/same/worse for adequacy and fluency in comparison to reference
- Excluding ties, agreement was 96% for adequacy and 95% for fluency
- All differences in judgments for the two systems were **highly significant** ($p < 0.001$, sign test)



Judges greatly preferred LSTM system on ±complete set



Judges also preferred LSTM system on =complete set, if not the same

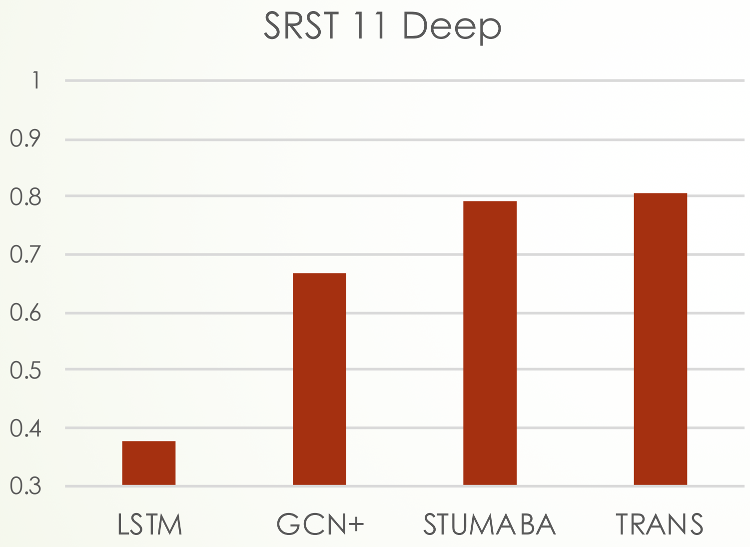
Related Work

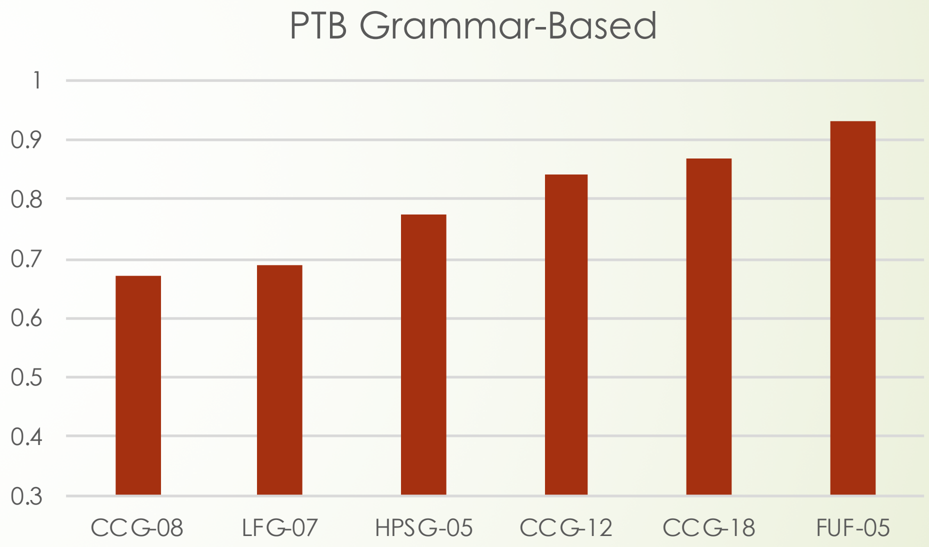
- LSTM hypertagging can potentially benefit other grammar-based methods using lexicalized grammars, e.g. using **HPSG** (Velldal and Oepen, 2005; Carroll and Oepen, 2005; Nakanishi et al., 2005) or **TAG** (Gardent and Perez-Beltrachini, 2017)
- Ok — but hasn't the field moved on to **end-to-end neural** methods? (Wen et al., 2015; Dušek and Jurcicek, 2016; Mei et al., 2016; Kiddon et al., 2016; Konstas et al., 2017; Wiseman et al., 2017)
- Maybe, but NNLG
  - is **difficult to control** and understand
  - often yields **incomplete** outputs and sometimes **hallucinates** content
  - has **not** been shown to work **better on complex texts** as in news genre

A surprising result? Old school HPSG parsing still beats neural parsing on DeepBank

- **DeepBank** (Flickinger et al., 2012) is a conversion of the Penn Treebank to Minimal Recursion Semantics (Copestake et al., 2005, MRS)
  - DeepBank ≈ OpenCCG semantic graphs ≈ SRST 2011 deep inputs
- For parsing, Buys and Blunsom (2017) found that their incremental **neural** semantic graph parser **lags 4-6% behind** an HPSG parser using a simple log-linear model (Toutanova et al., 2005) on DeepBank
  - HPSG parser ≫ B&B (2017) ≫ attentional seq2seq

Grammar-based deep realization may still exceed neural as well





- LSTM (Marcheggiani & Perez-Beltrachini, 2018), (M&P-B, 2018), (Bohnet et al., 2011), (Zhang et al., 2017)
- CCG-08 (White et al., 2008), LFG (Hogan et al., 2007), HPSG (Nakanishi et al., 2005), CCG-12 (White & Rajkumar, 2012), CCG-18 (this paper), FUF-05 (Callaway, 2005)

Next steps: Directly compare neural and grammar-based methods

- Of course, inputs to grammar-based systems are only (roughly?) comparable to shared task deep inputs
- Could try Marcheggiani & Perez-Beltrachini's (2018) neural method on inputs to grammar-based systems!
- Also important to look at performance when augmenting training data with auto-parsed inputs (Elder & Hokamp, 2018)
- And can try neuralizing dependency-based (Song et al., 2018) and grammar-based approaches!

Conclusions

- We have shown that our **LSTM hypertagger** significantly **outperforms** the existing maxent OpenCCG hypertagger **on** both **tagging accuracy** and downstream **realization** performance
- Important role of input linearization suggests looking at graph convolutional networks for hypertagging
- Neuralizing realization ranker can be expected to yield further gains

Thanks

- to the OSU Clippers Group, Alan Ritter and the anonymous reviewers for helpful comments and discussion
- to Sarah Ewing and Amad Hussain for their assistance with the evaluation
- to the US National Science Foundation, the Ohio Supercomputer Center and Facebook
- and to **YOU** for listening!