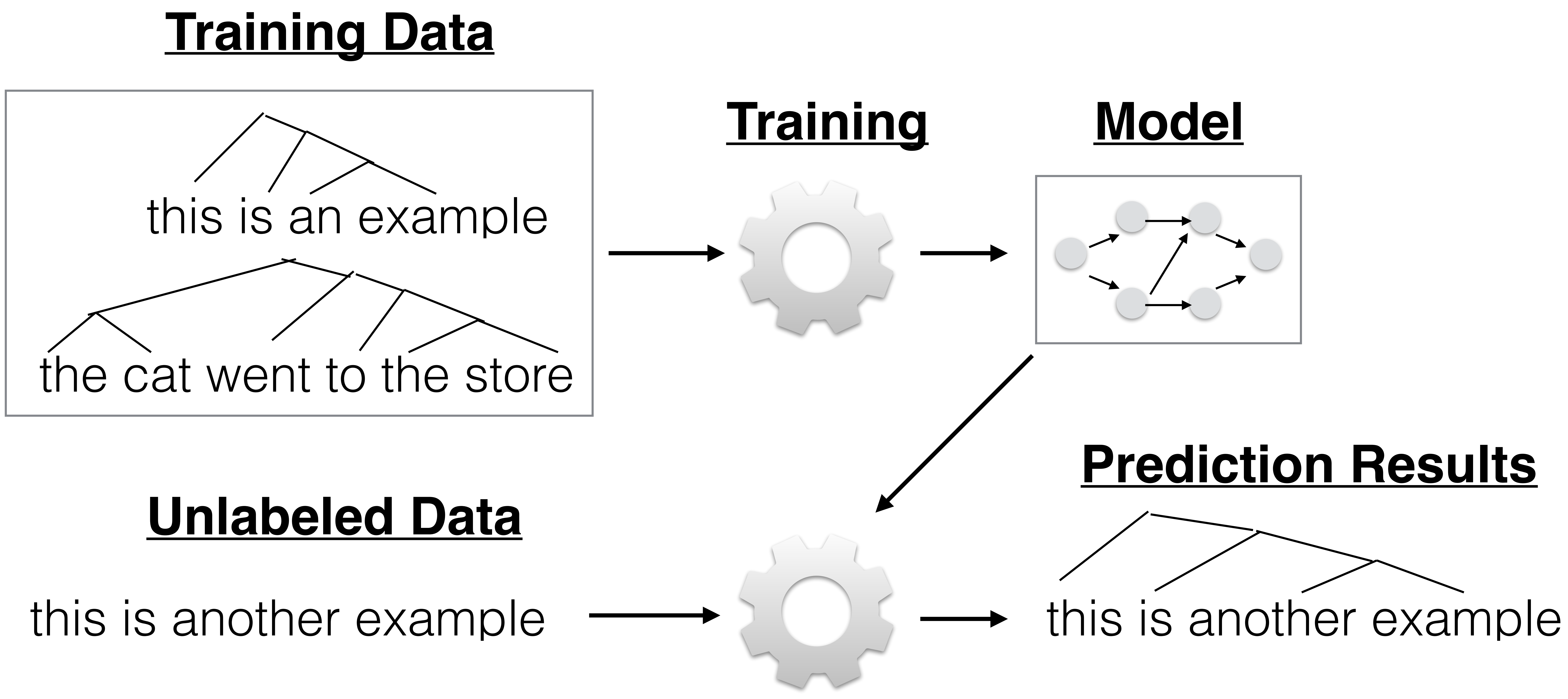
What Can Neural Networks Teach Us about Language?

Graham Neubig

Carnegie Mellon University
Language Technologies Institute

@ New York University 4/12/2018

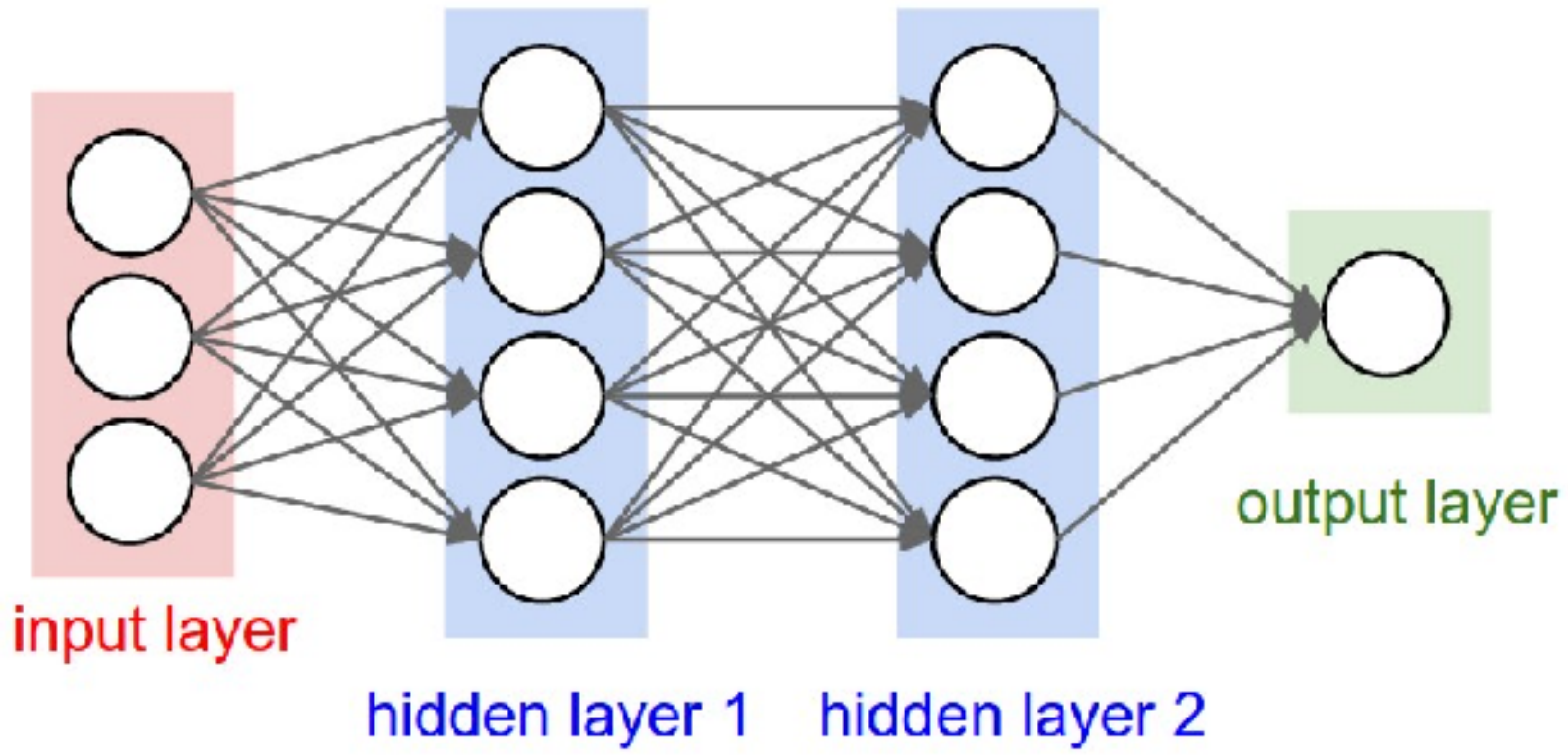
Supervised Training for Natural Language Processing



Neural networks are mini-scientists!

Syntax?

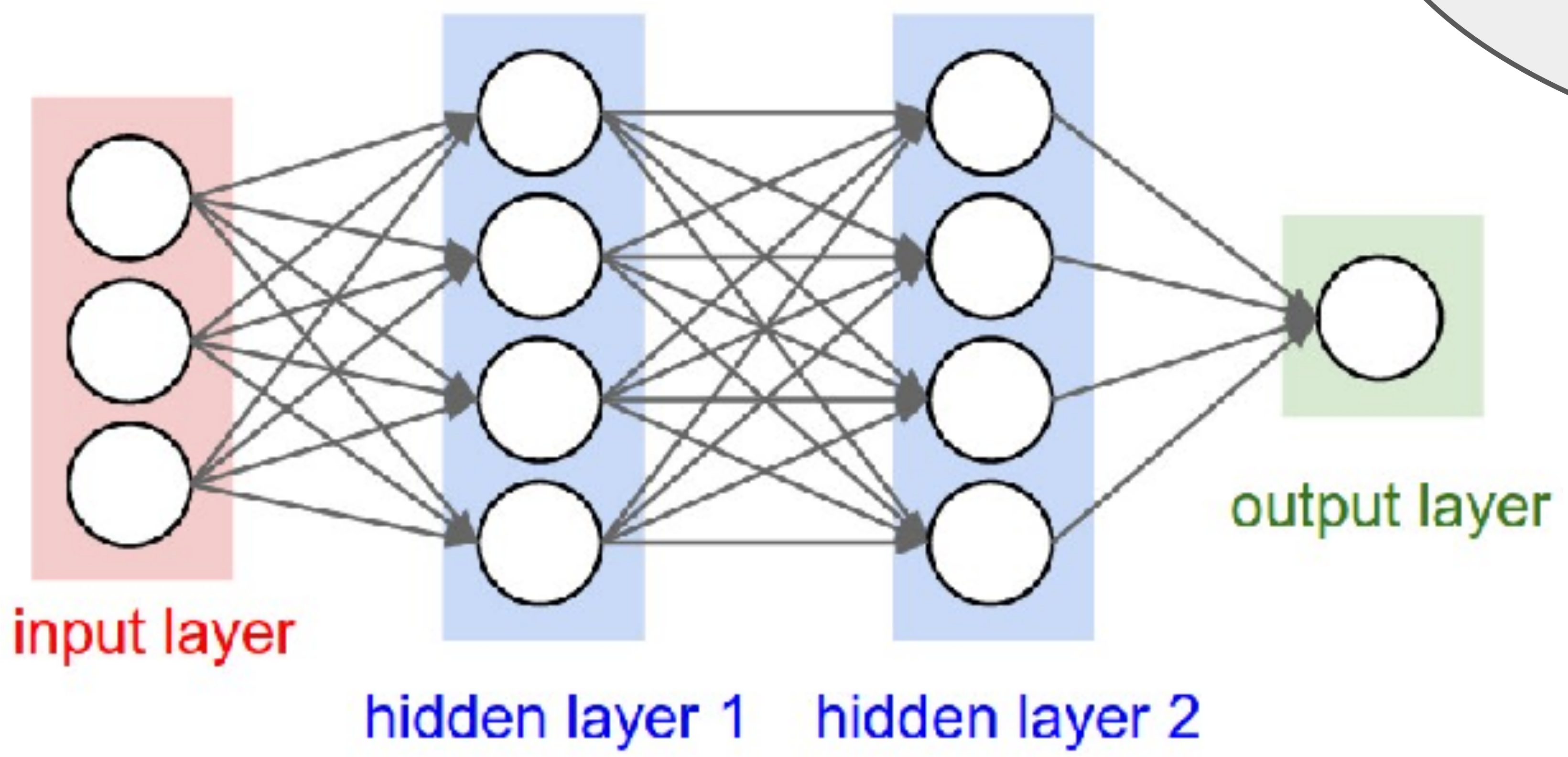
Semantics?



Neural networks are mini-scientists!

Syntax?

Semantics?

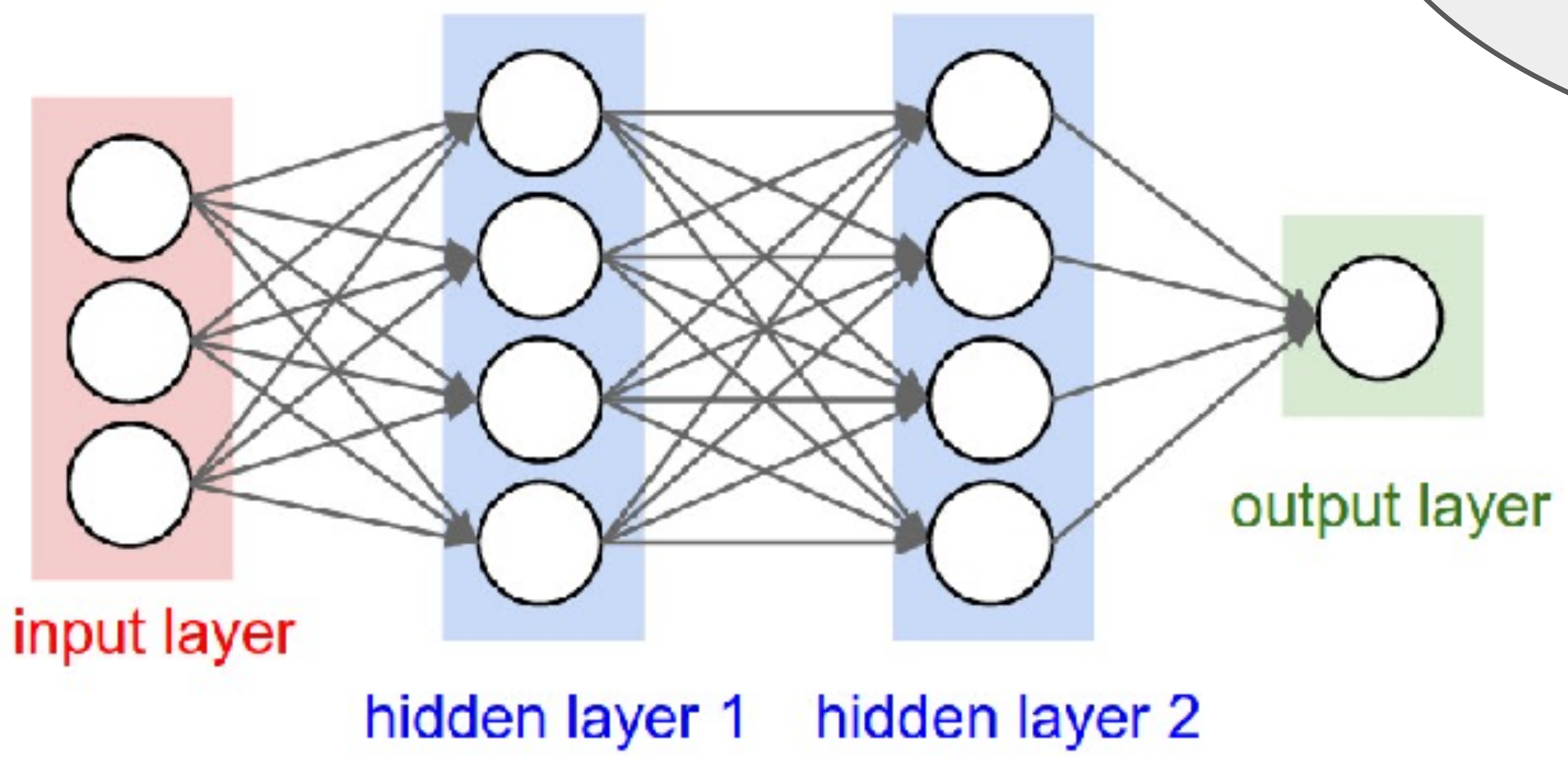


What syntactic phenomena do you learn?

Neural networks are mini-scientists!

Syntax?

Semantics?

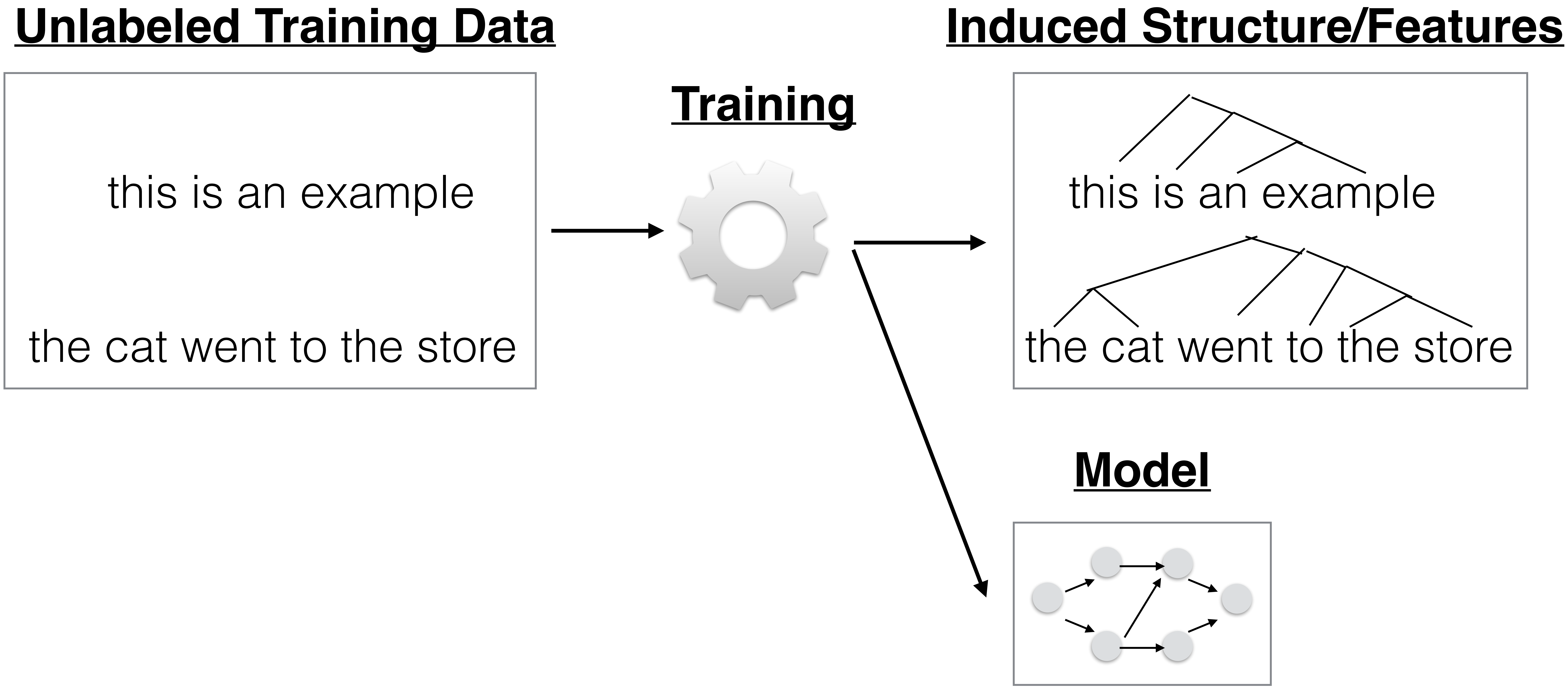


What syntactic phenomena do you learn?

New way of testing linguistic hypothesis

Basis to further improve the model

Unsupervised Training of Neural Networks for Language



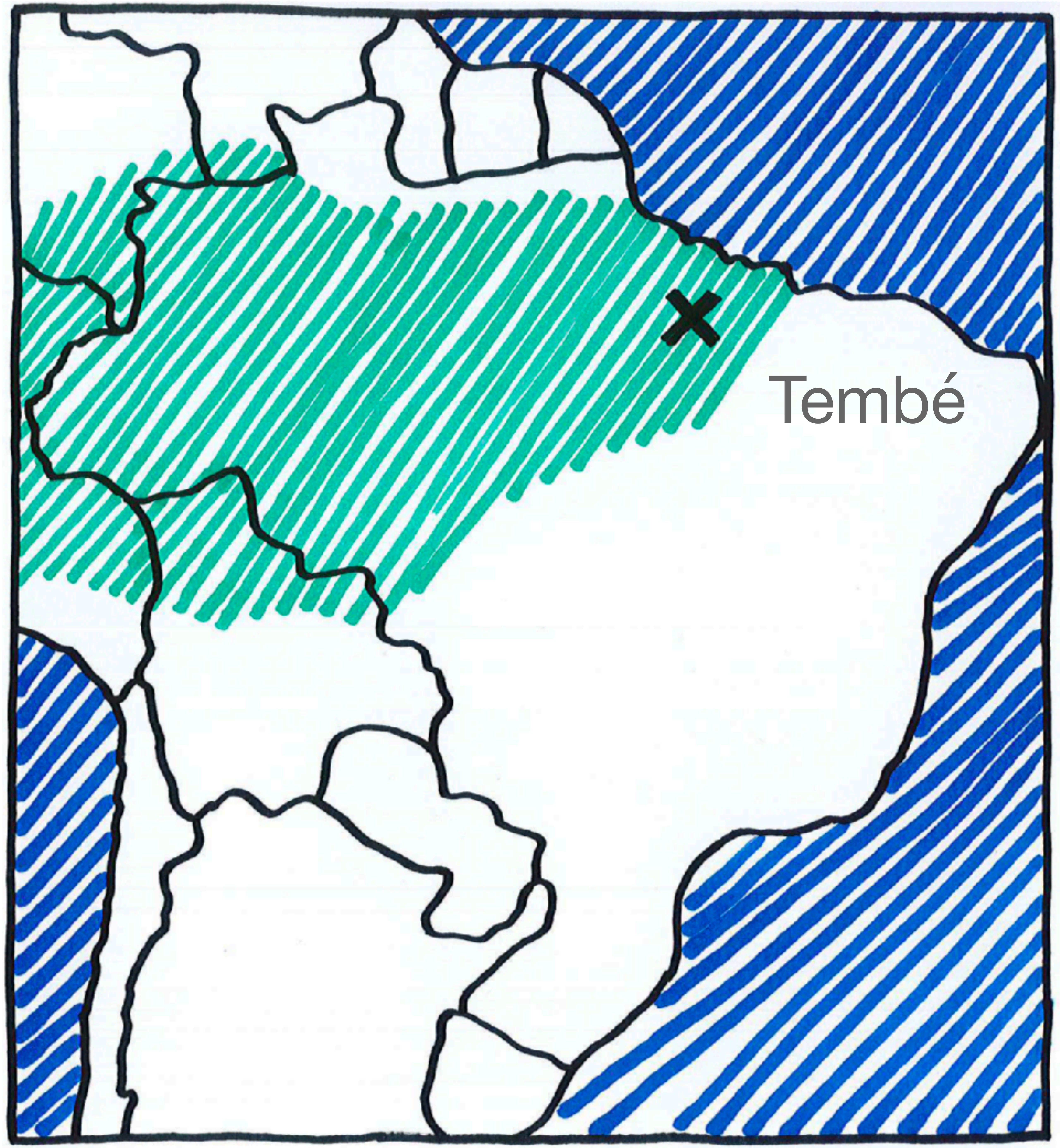
Three Case Studies

- Learning features of a language through translation
- Learning about linguistic theories by learning to parse
- Methods to accelerate your training for NLP and beyond

Learning Language Representations for Typology Prediction

Chaitanya Malaviya, Graham Neubig, Patrick Littell
EMNLP2017

http://endangeredlanguages.com/


Tembé

Augustine

Emidio

Photos by Steven Bird

Why Document Endangered Languages?

- **For young speakers:** in many cultures, revived interest in learning their ancestral language.
- **For posterity:** our incredibly rich linguistic heritage is in danger, and at the very least, we'd like to preserve it.

Linguistic Typology

Syntax: e.g. what is the word order?

English = SVO: *he bought a car* **Japanese** = SOV: *kare wa kuruma wo katta*

Irish = VSO: *cheannaigh sé carr* **Malagasy** = VOS: *nividy fiara izy*

Morphology: e.g. how does it conjugate words?

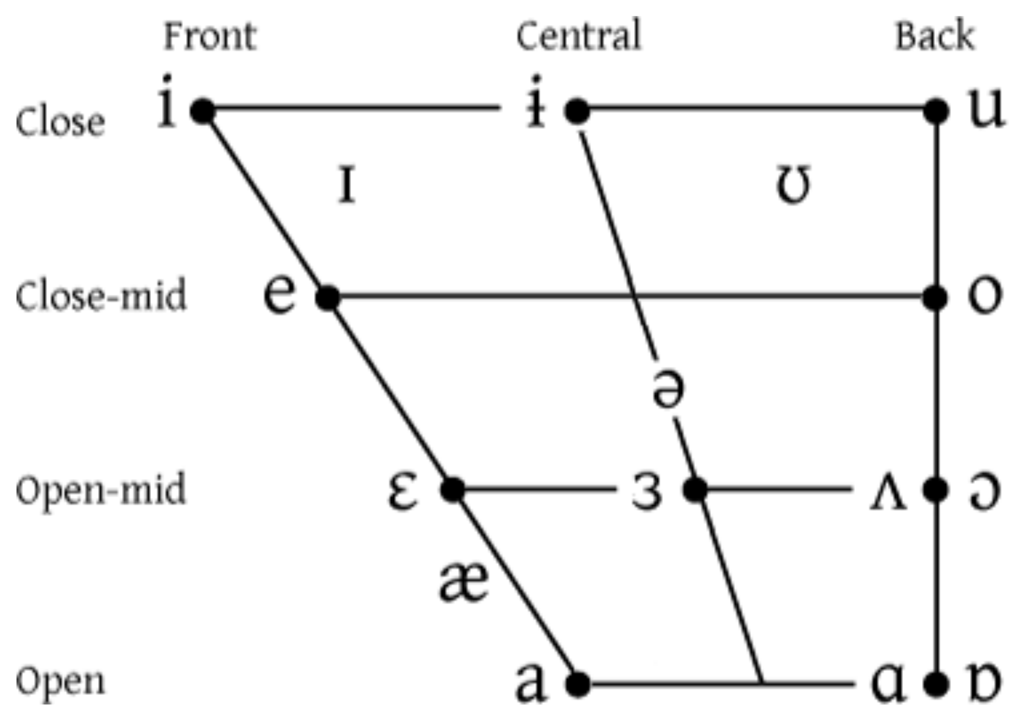
English = fusional: *she opened the door for him again*

Japanese = agglutinative: *kare ni mata doa wo aketeageta*

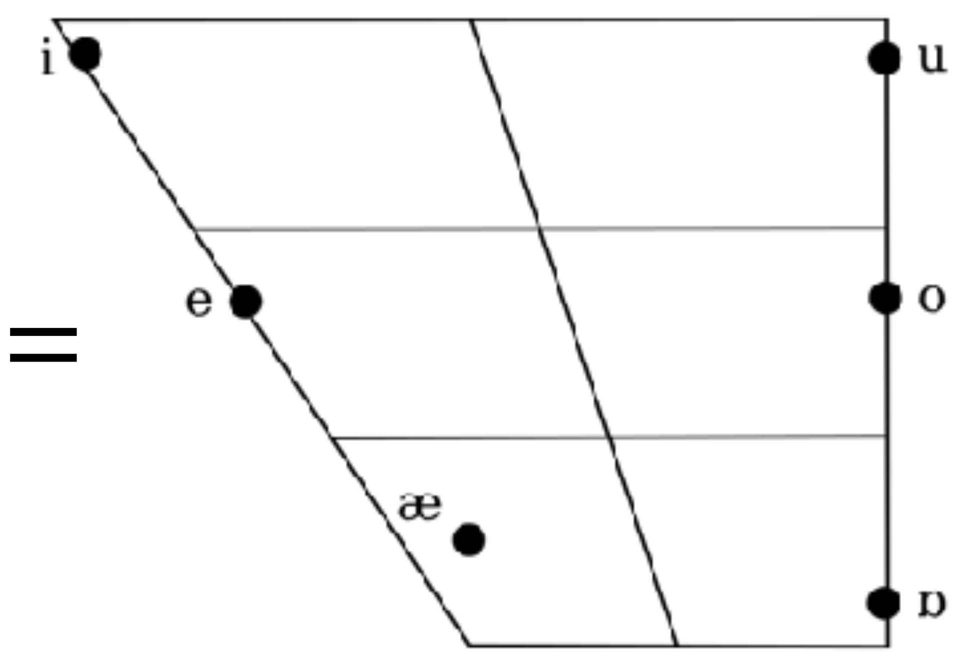
Mohawk = polysynthetic: *sahonwanhotónkwahse*

Phonology: e.g. what is its inventory of vowel sounds?

English =



Farsi =



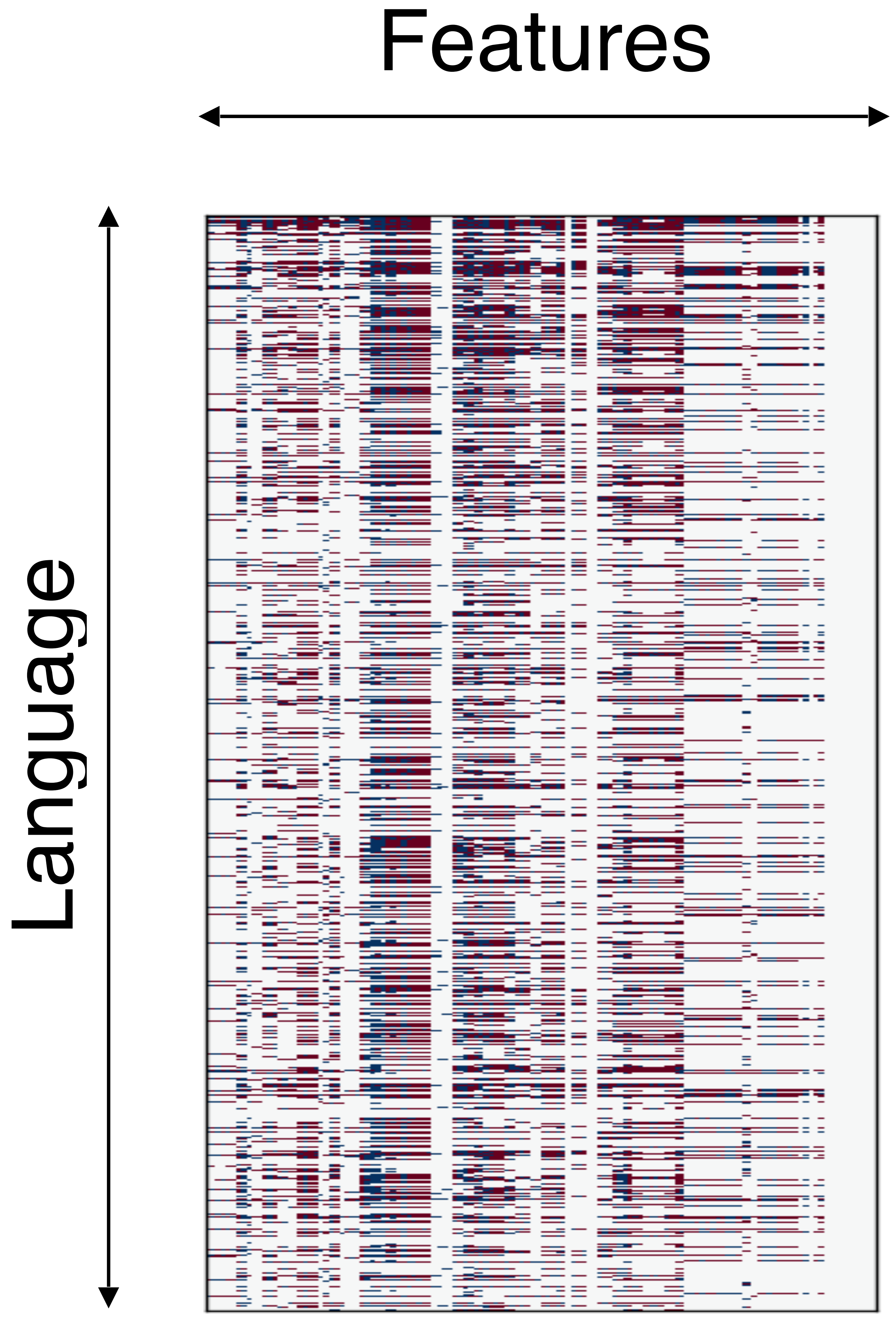
“Encyclopedias” of Linguistic Typology

- There are 7,099 living languages in the world
- Databases that contain information about their features
  - World Atlas of Language Structures (Dryer & Haspelmath 2013)
  - Syntactic Structures of the World’s Languages (Collins & Kayne 2011)
  - PHOIBLE (Moran et al. 2014)
  - Ethnologue (Paul 2009)
  - Glottolog (Hammarström et al. 2015)
  - Unicode Common Locale Data Repository, etc.

Information is Woefully Incomplete!

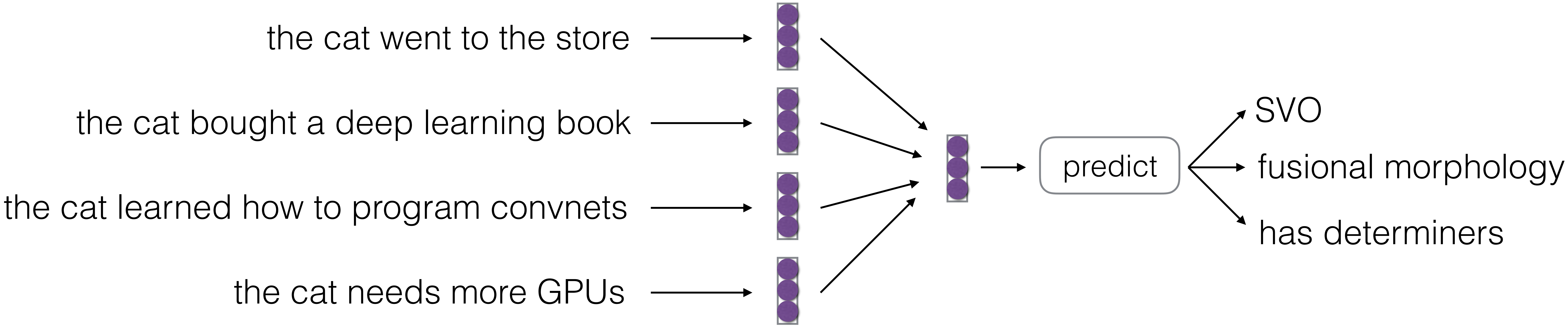
- The *World Atlas of Language Structures* is a general database of typological features, covering ≈200 topics in ≈2,500 languages.
- Of the possible feature/value pairs, only about 15% have values

- Can we **learn** to fill in this missing knowledge about the languages of the world?



How Do We Learn about an Entire Language?!

- Proposed Method:
  - Create representations of each sentence in the language
  - Aggregate the representations over all the sentences
  - Predict the language traits



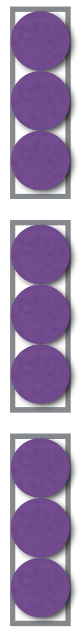
How do we Represent Sentences?

- Our proposal: learn a multi-lingual translation model

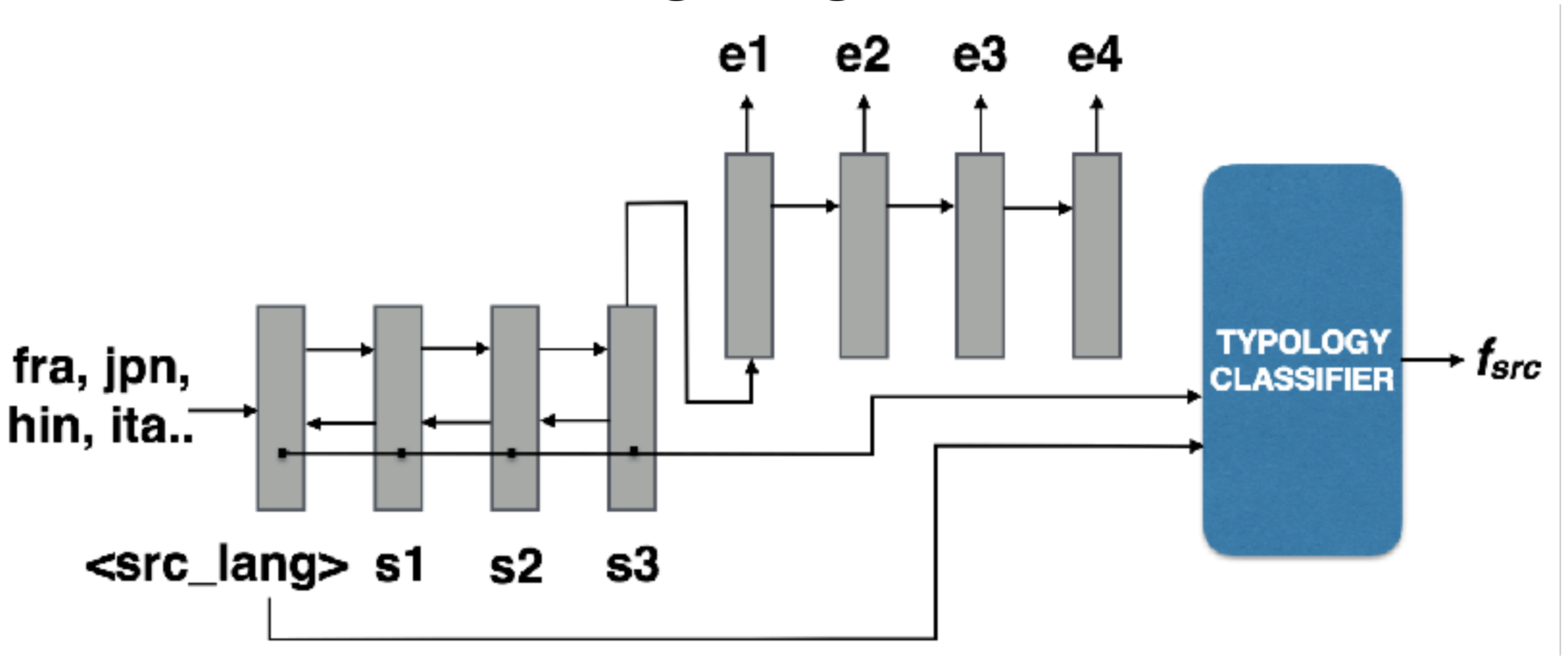
<Japanese> *kare wa kuruma wo katta* → → *he bought a car*

<Irish> *cheannaigh sé carr* → → *he bought a car*

<Malagasy> *nividy fiara izy* → → *he bought a car*



- Extract features from the language token and intermediate hidden states



- Inspired by previous work that demonstrated that MT hidden states have correlation w/ syntactic features (Shi et al. 2016, Belinkov et al. 2017)

Experiments

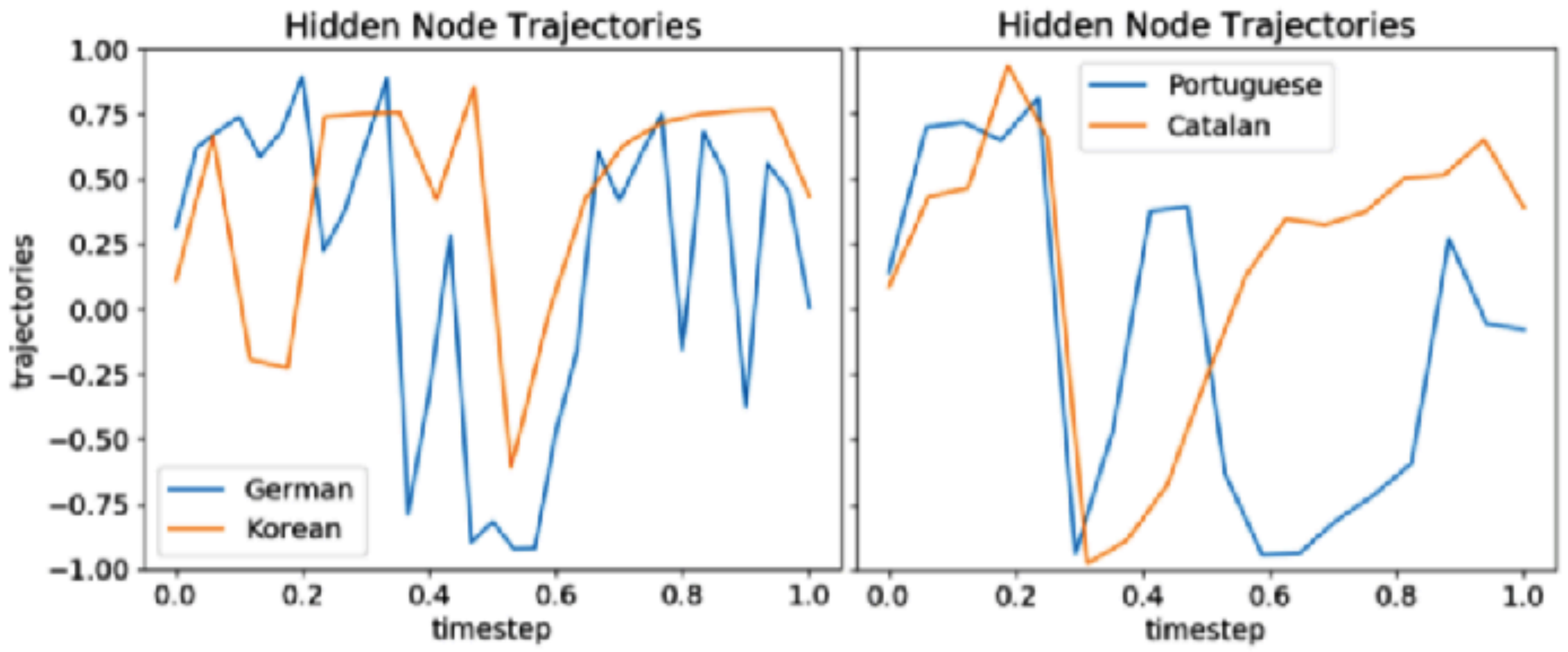
- Train an MT system translating 1017 languages to English on text from the Bible
- Learned language vectors **available here**: https://github.com/chaitanyamalaviya/lang-reps
- Estimate typological features from the URIEL database (http://www.cs.cmu.edu/~dmortens/uriel.html) using cross-validation
- **Baseline:** a k-nearest neighbor approach based on language family and geographic similarity

Results

- Learned representations encode information about the entire language, and help w/ predicting its traits (c.f. language model)

	Syntax		Phonology		Inventory	
	-Aux	+Aux	-Aux	+Aux	-Aux	+Aux
NONE	69.91	83.07	77.92	86.59	85.17	90.68
LMVEC	71.32	82.94	80.80	86.74	87.51	89.94
MTVEC	74.90	83.31	82.41	87.64	89.62	90.94
MTCELL	75.91	85.14	84.33	88.80	90.01	90.85
MTBOTH	**77.11**	**86.33**	**85.77**	**89.04**	**90.06**	**91.03**

- Trajectories through the sentence are similar for similar languages



GER: Ich bin das A und das O , der Anfang und das Ende , spricht Gott der HERR , der da ist und der da war und der da kommt , der Allmächtige .

KOR: 지금도 계시고 전에도 계셨고 앞으로 오실 전능하신 주 하나님께서 " 나는 알파요 오메가다 " 하고 말씀하십니다 .

POR: Paulo , chamado para ser apóstolo de Cristo Jesus pela vontade de Deus , e o irmão Sóstenes

CAT: Pau , cridat per voler de Déu a ser apòstol de Jesucrist , i el germà Sòstenes

We Can Learn About Language from Unsupervised Learning!

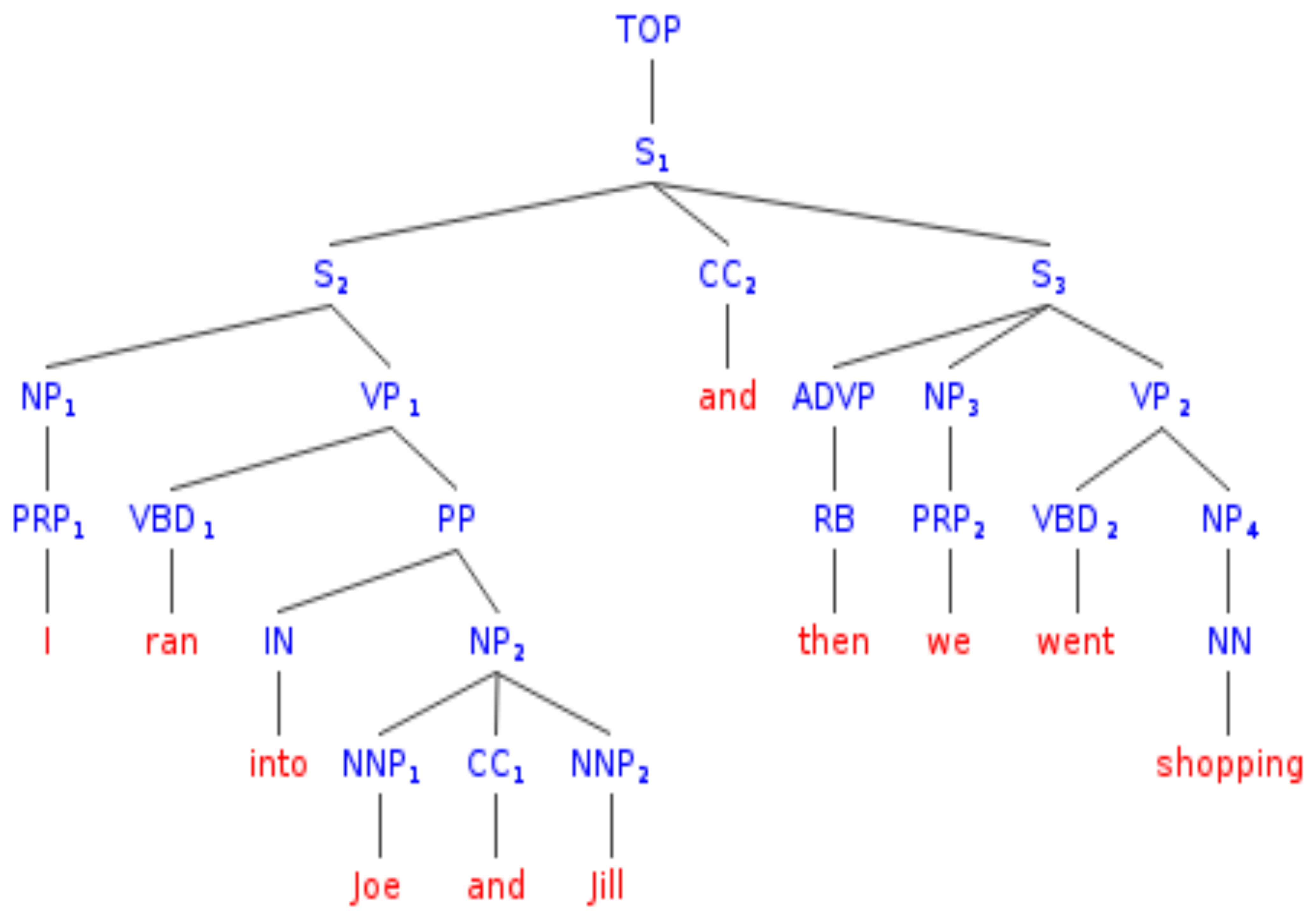
- We can use deep learning and naturally occurring translation data to learn features of language as a whole.
- But this is still on the level of extremely coarse-grained typological features
- What if we want to examine specific phenomena in a deeper way?

What Can Neural Networks Learn about Syntax?

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong
Chris Dyer, Graham Neubig, Noah A. Smith
EACL2017 (Outstanding Paper Award)

An Alternative Way of Generating Sentences

P(***x***)



P(***x***, ***y***)

Overview

- Crash course on Recurrent Neural Network Grammars (RNNG)
- Answering linguistic questions through RNNG learning

Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

No. Steps	Stack	String Terminals	Action
0			NT(S)
1	(S		NT(NP)

Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

No. Steps	Stack	Terminals	Action
0			NT(S)
1	(S		NT(NP)
2	(S | (NP		GEN(*the*)

Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

No. Steps	Stack	Terminals	Action
0			NT(S)
1	(S		NT(NP)
2	(S | (NP		GEN(*the*)
3	(S | (NP | *the*	*the*	GEN(*hungry*)

Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

No. Steps	Stack	Terminals	Action
0			NT(S)
1	(S		NT(NP)
2	(S | (NP		GEN(*the*)
3	(S | (NP | *the*	*the*	GEN(*hungry*)
4	(S | (NP | *the* | *hungry*	*the hungry*	GEN(*cat*)

Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

No. Steps	Stack	Terminals	Action
0			NT(S)
1	(S		NT(NP)
2	(S | (NP		GEN(*the*)
3	(S | (NP | *the*	*the*	GEN(*hungry*)
4	(S | (NP | *the* | *hungry*	*the hungry*	GEN(*cat*)
5	(S | (NP | *the* | *hungry* | *cat*	*the hungry cat*	REDUCE

Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

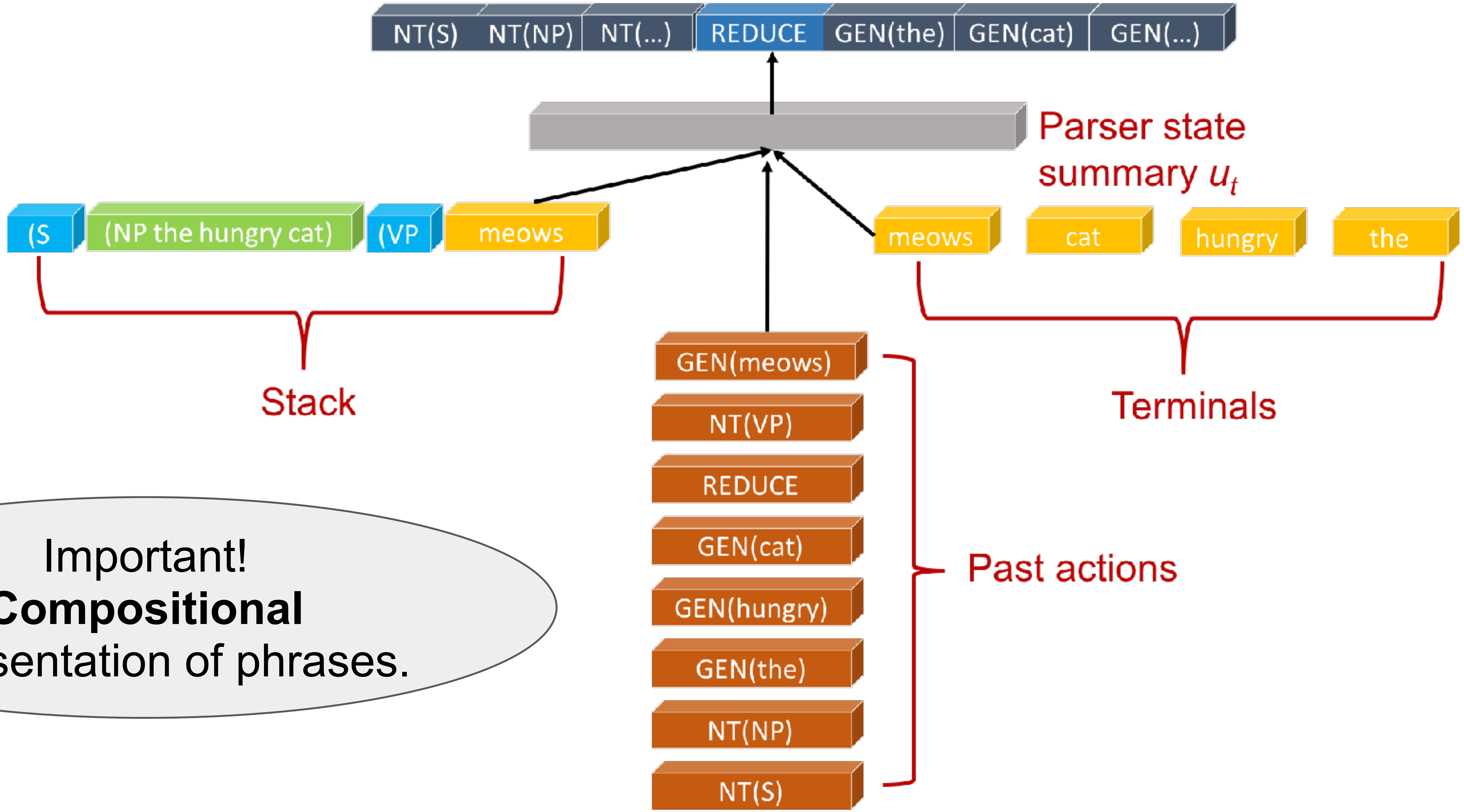
No. Steps	Stack	Terminals	Action
0			NT(S)
1	(S		NT(NP)
2	(S | (NP		GEN(*the*)
3	(S | (NP | *the*	*the*	GEN(*hungry*)
4	(S | (NP | *the* | *hungry*	*the hungry*	GEN(*cat*)
5	(S | (NP | *the* | *hungry* | *cat*	*the hungry cat*	REDUCE
6	(S | (NP *the hungry cat*)	*the hungry cat*	NT(VP)

Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

No. Steps	Stack	Terminals	Action
0			NT(S)
1	(S		NT(NP)
2	(S | (NP		GEN(*the*)
3	(S | (NP | *the*	*the*	GEN(*hungry*)
4	(S | (NP | *the* | *hungry*	*the hungry*	GEN(*cat*)
5	(S | (NP | *the* | *hungry* | *cat*	*the hungry cat*	REDUCE
6	(S | (NP *the hungry cat*)	*the hungry cat*	NT(VP)

Model Architecture



Important!
Compositional
representation of phrases.

PTB Test Experimental Results

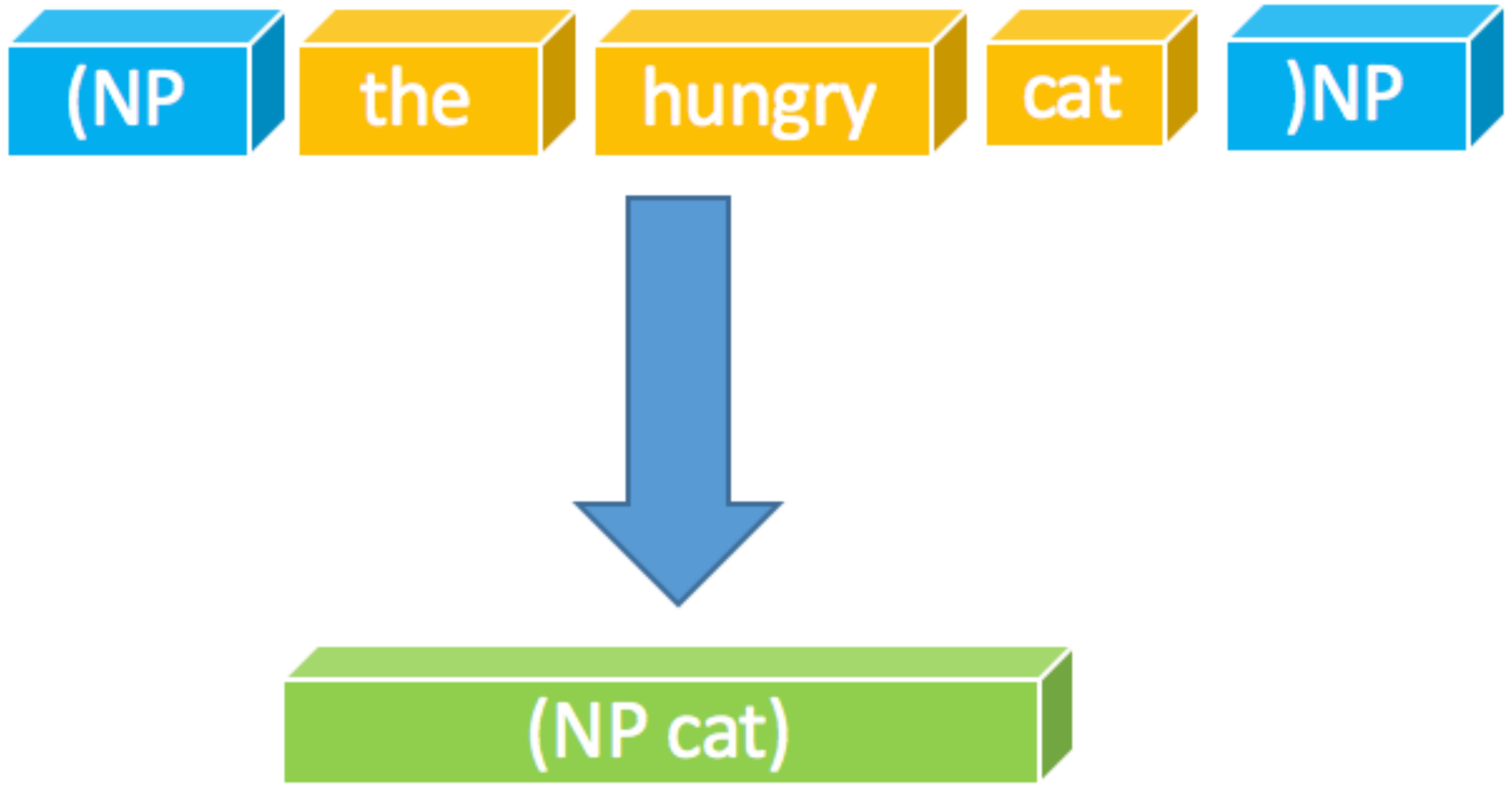
Parsing F1

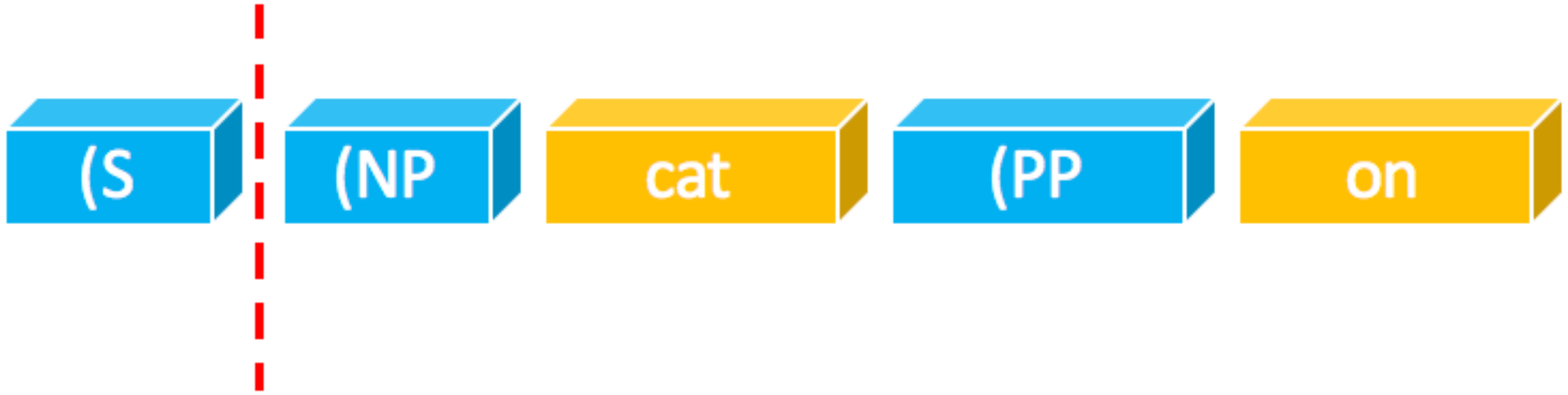
Model	Parsing F1
Collins (1999)	88.2
Petrov and Klein (2007)	90.1
RNNG	**93.3**
Choe and Charniak (2016) - Supervised	92.6

LM Ppl.

Model	LM ppl.
IKN 5-gram	169.3
Sequential LSTM LM	113.4
RNNG	**105.2**

In The Process of Learning, Can RNNGs Teach Us About Language?





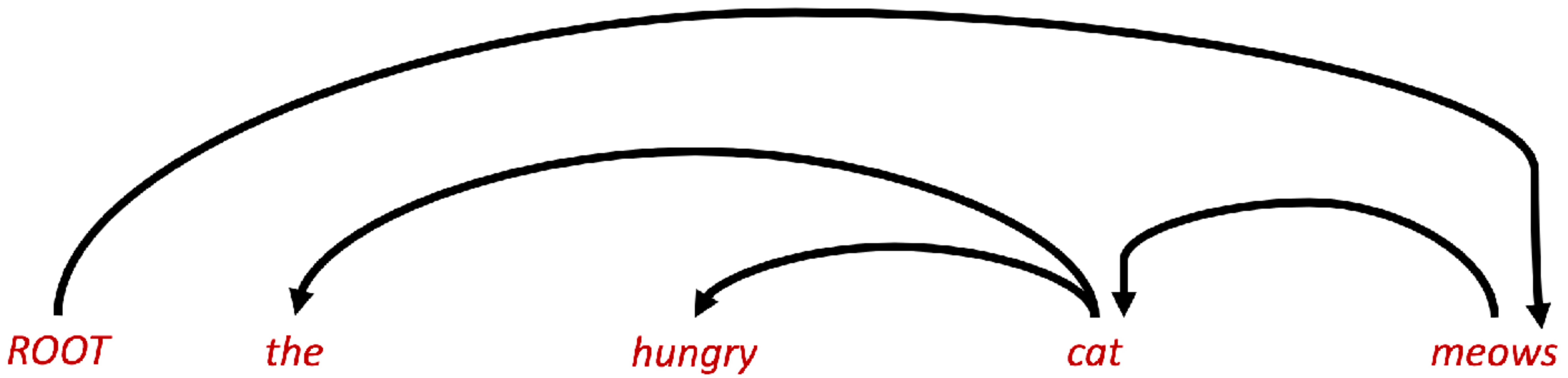
Lexicalization

Parent annotations

Question 1: Can The Model Learn "Heads"?

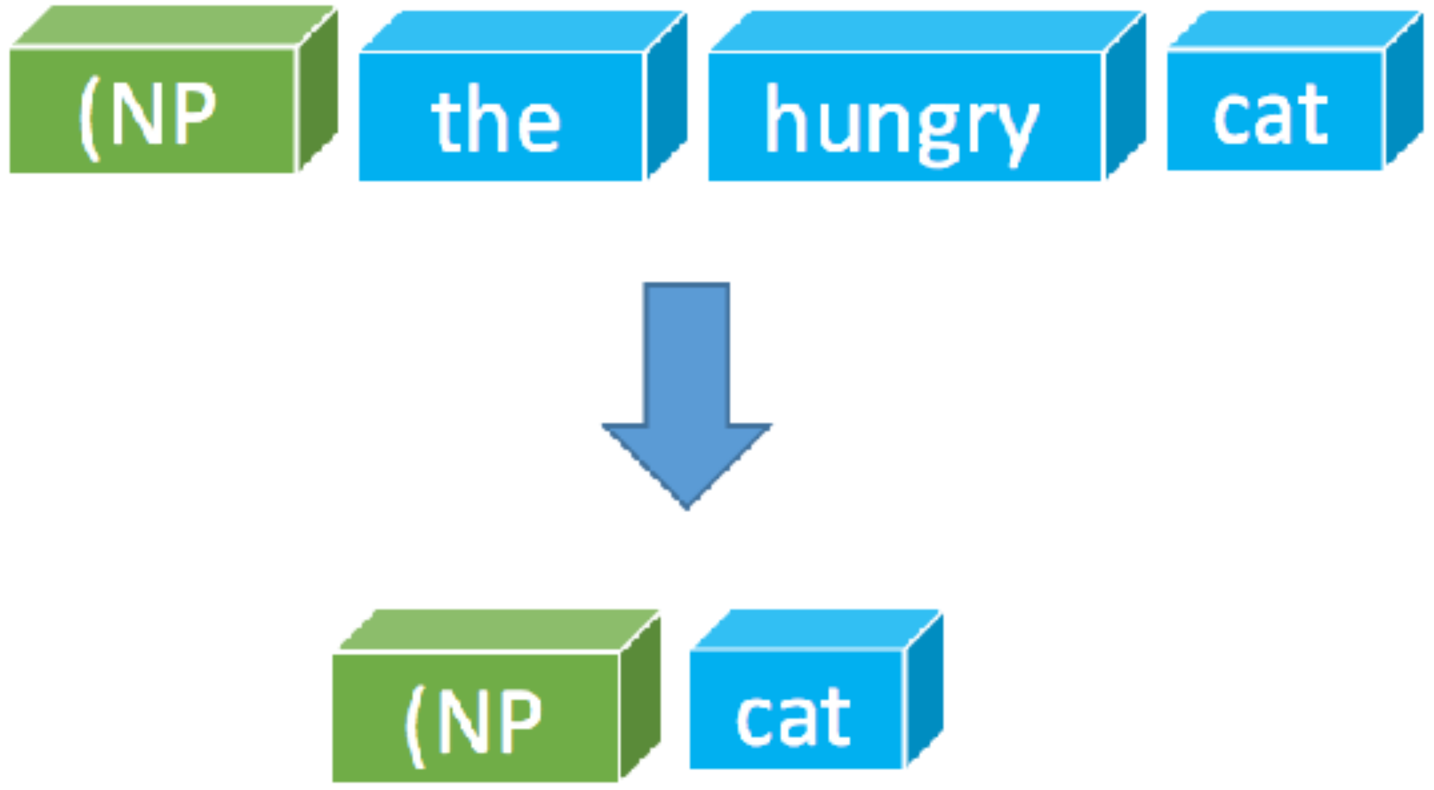
Method: New interpretable attention-based composition function

Result: sort of



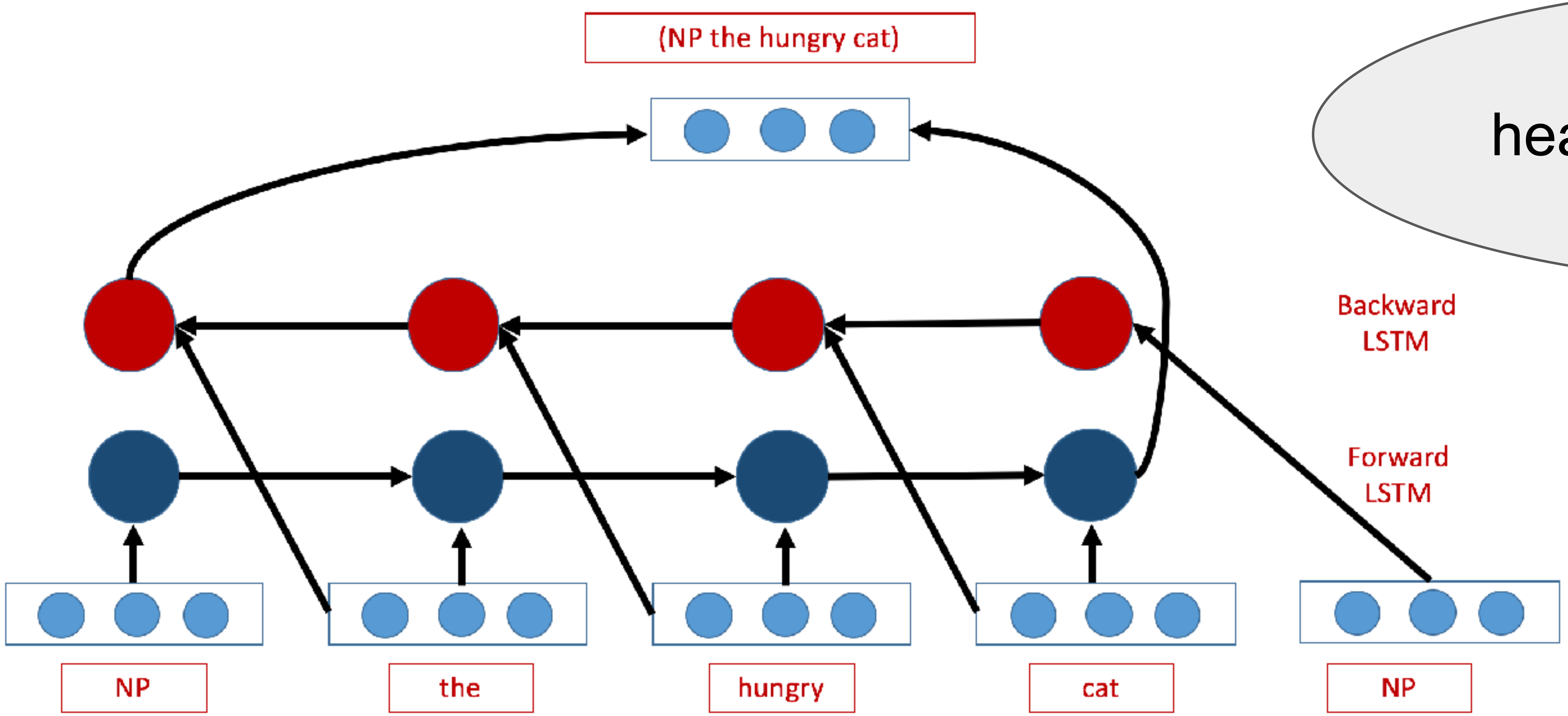
Headedness

- Linguistic theories of phrasal representation involve a strongly privileged lexical head that determines the whole representation



- Hypothesis for single lexical heads (Chomsky, 1993) and multiple ones for tricky cases (Jackendoff 1977; Keenan 1987)

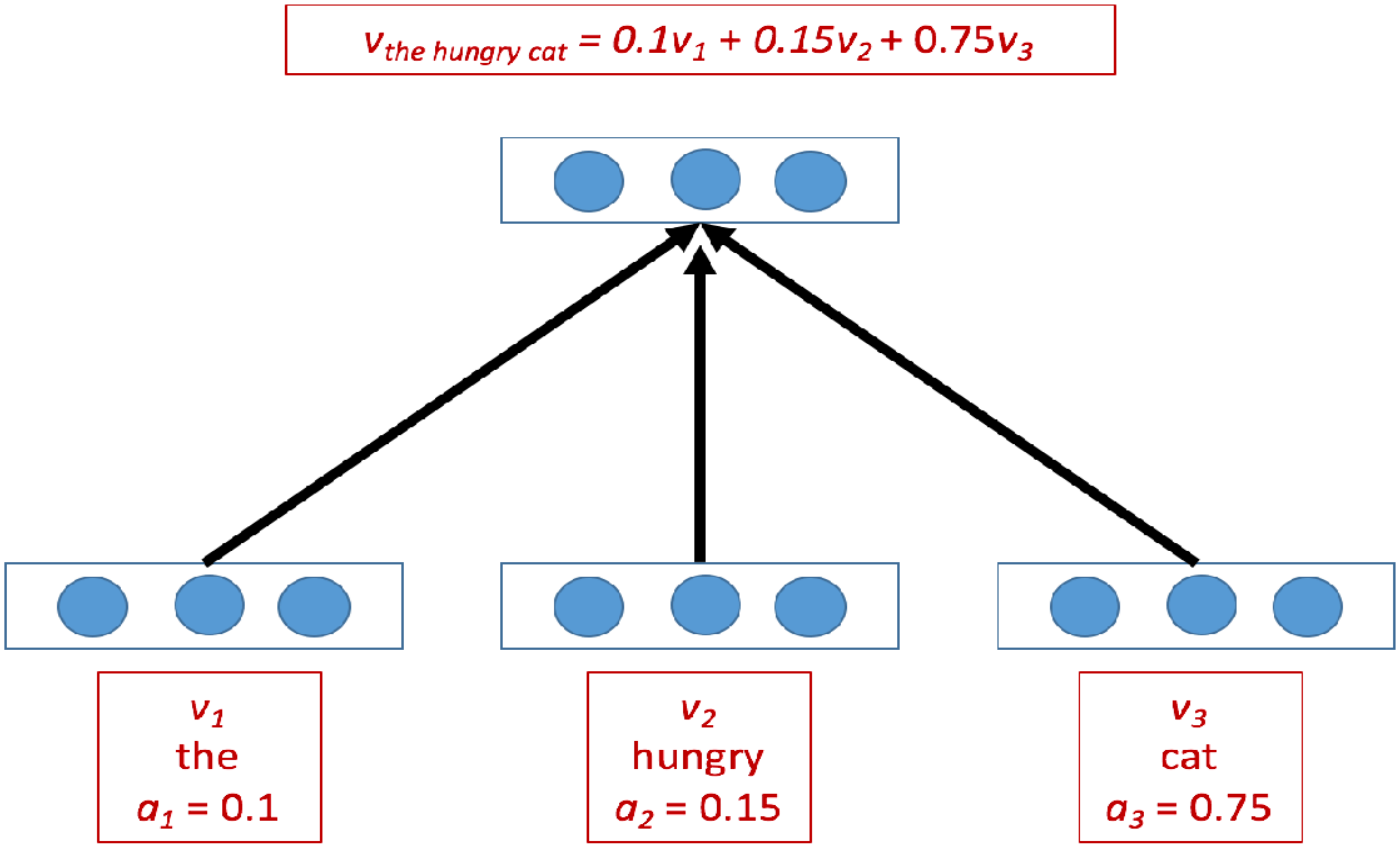
RNNG Composition Function



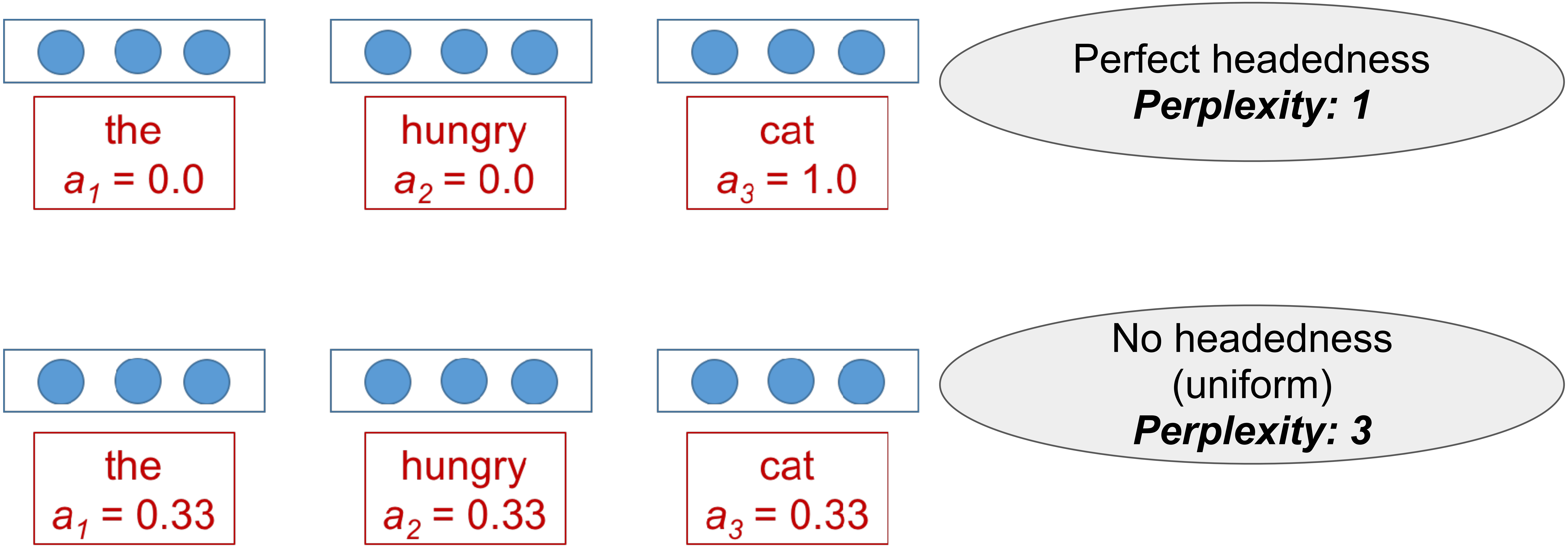
Hard to detect headedness in sequential LSTMs

Use “attention” in sequence-to-sequence model (Bahdanau et al., 2014)

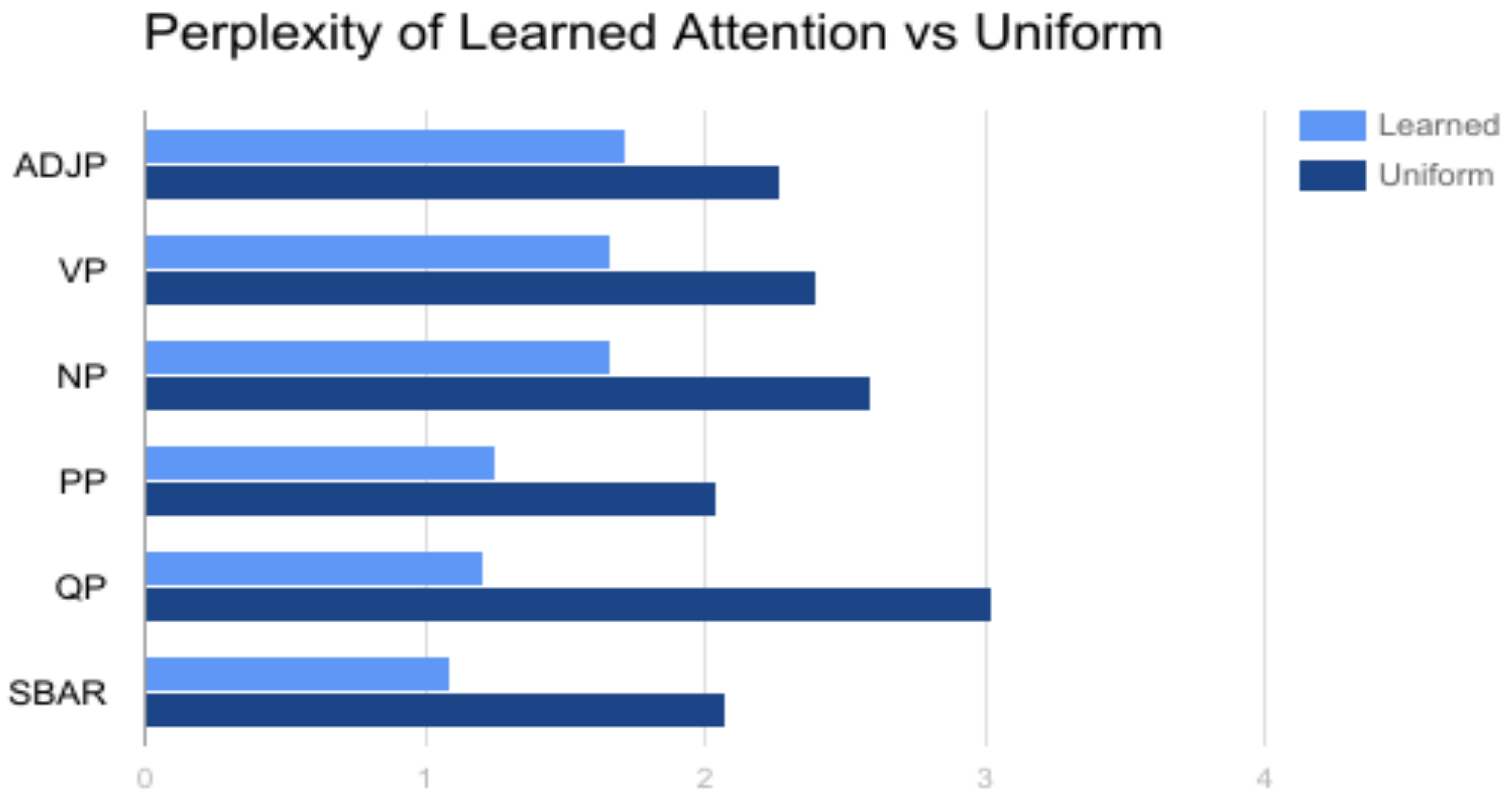
Key Idea of Attention



Two Extreme Cases of Attention



Perplexity of the Attention Vectors



Learned Attention Vectors

Noun Phrases
the (0.0) final (0.18) **hour (0.81)**
their (0.0) first (0.23) **test (0.77)**
Apple (0.62) , (0.02) Compaq (0.1) and (0.01) IBM (0.25)
NP (0.01) , (0.0) **and (0.98)** NP (0.01)

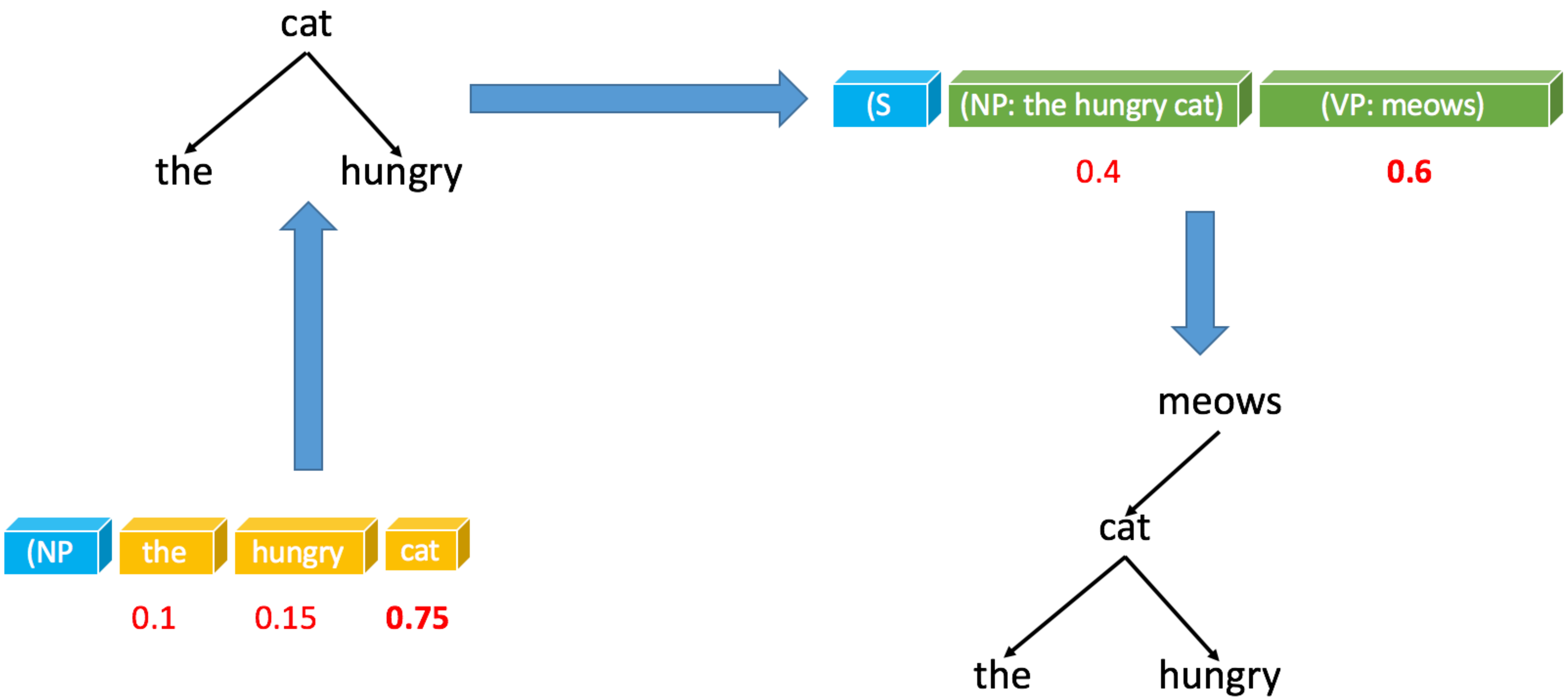
Learned Attention Vectors

Verb Phrases
to (0.99) VP (0.01)
did (0.39) **n’t (0.60)** VP (0.01)
handle (0.09) **NP (0.91)**
VP (0.15) **and (0.83)** VP (0.02)

Learned Attention Vectors

Prepositional Phrases
of (0.97) NP (0.03)
in (0.93) NP (0.07)
by (0.96) S (0.04)
NP (0.1) **after** (0.83) NP (0.06)

Quantifying the Overlap with Head Rules



Quantifying the Overlap with Head Rules

Reference	UAS
Random baseline	~28.6
Collins head rules	49.8
Stanford head rules	40.4

Question 2: Can the Model Learn Phrase Types?

Method: Ablate the nonterminal label categories from the data

Result: Nonterminal labels add very little, and the model learns something similar automatically

Role of Nonterminals

- Exploring the endocentric or exocentric hypothesis of phrasal representation

Endocentric: represent an NP with the noun headword

Exocentric: *S* → *NP VP* (relabel *NP* and *VP* with a new syntactic category "*S*")

- We use a data ablation procedure by replacing all nonterminal symbols with a single nonterminal category "*X*"

Nonterminal Ablation

(S (NP the hungry cat) (VP meows) .)

(X (X the hungry cat) (X meows) .)

Quantitative Results

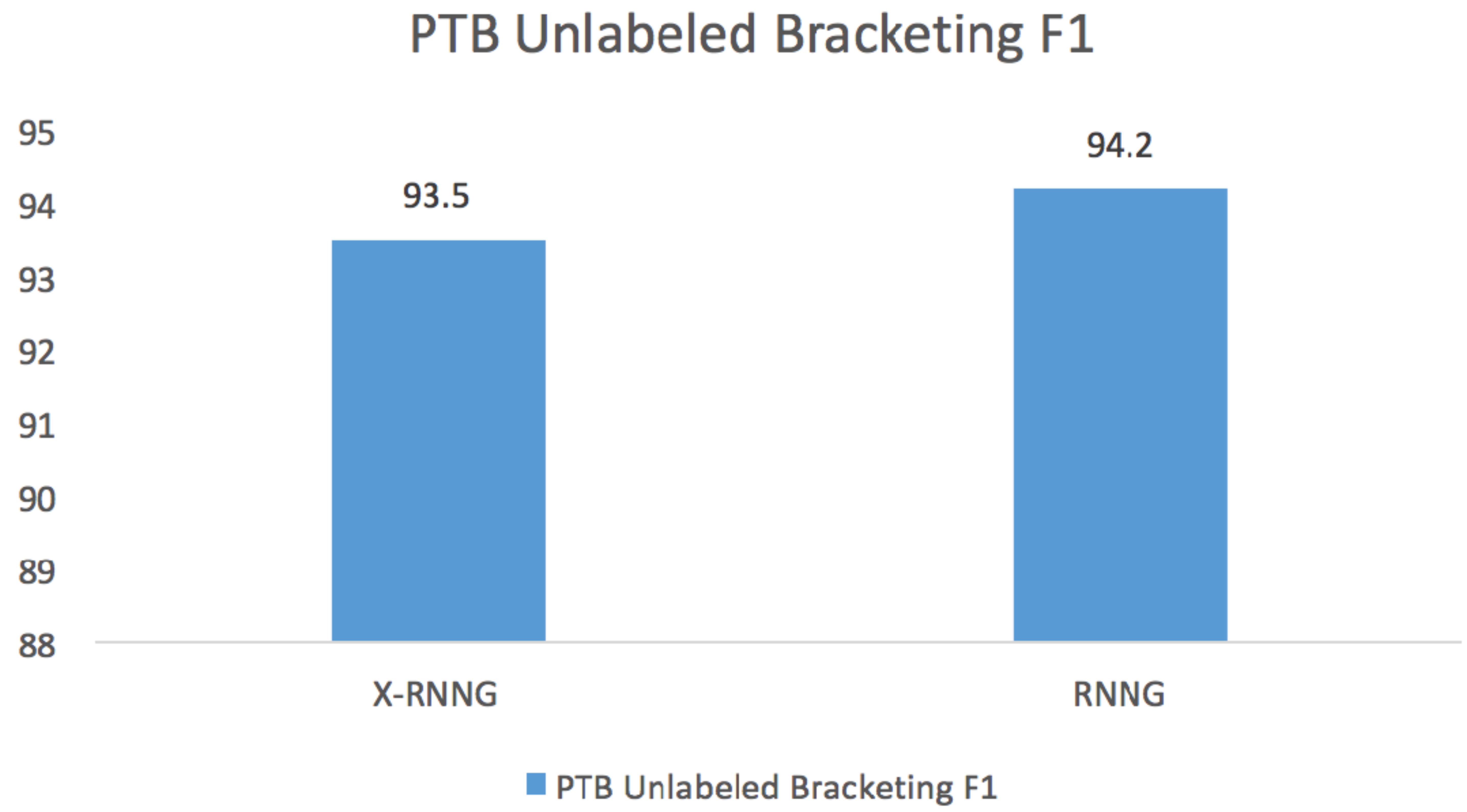
Gold: (X (X the hungry cat) (X meows) .)

Predicted: (X (X the hungry) (X cat meows) .)

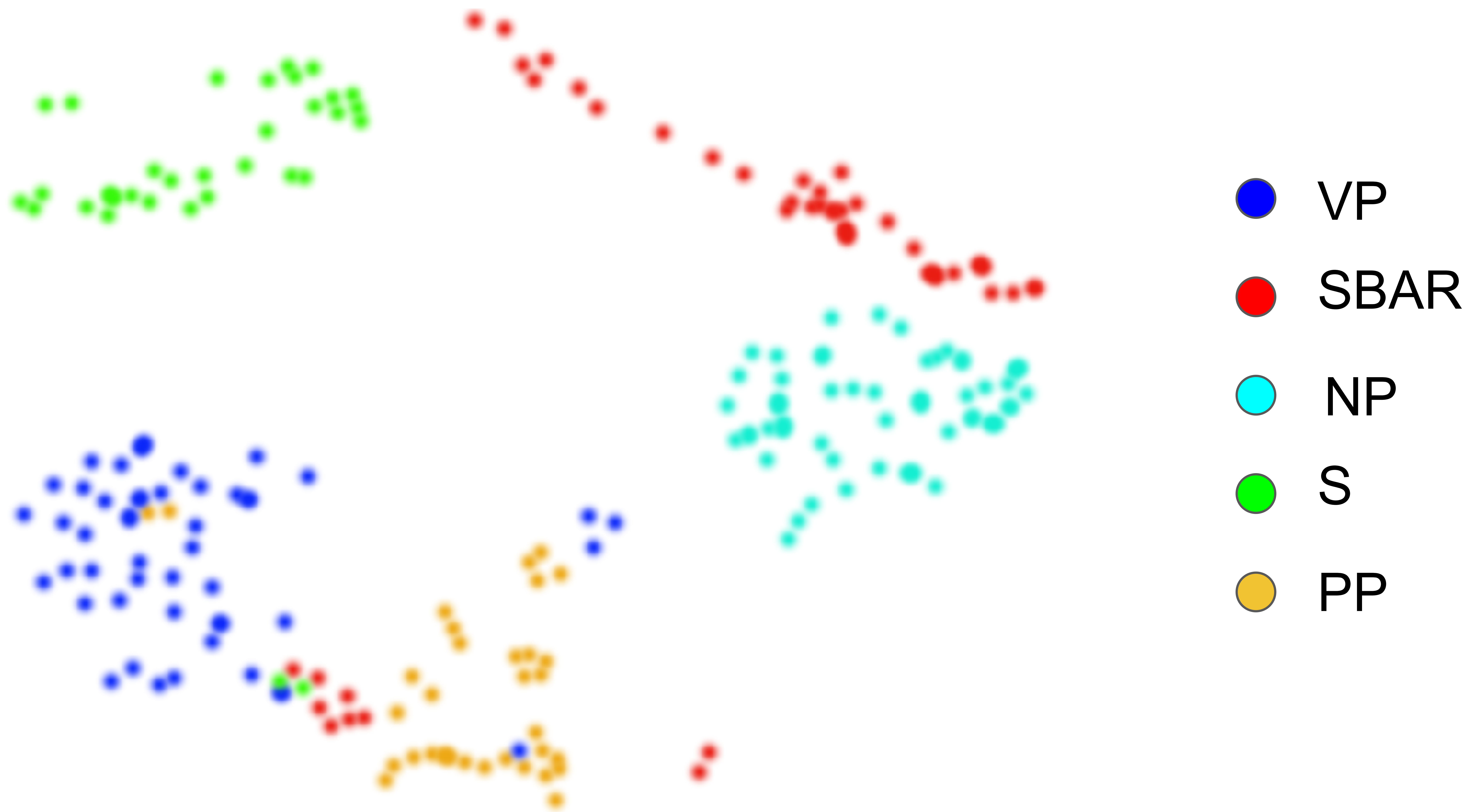
Quantitative Results

Gold: (X (X the hungry cat) (X meows) .)

Predicted: (X (X the hungry) (X cat meows) .)



Visualization



Conclusion

- RNNG learns (imperfect) headedness, which is both similar and distinct to linguistic theories
- RNNG is able to rediscover nonterminal information given weak bracketing structures, and also make nontrivial semantic distinctions

On-the-fly Operation Batching in Dynamic Computation Graphs

Graham Neubig, Yoav Goldberg, Chris Dyer

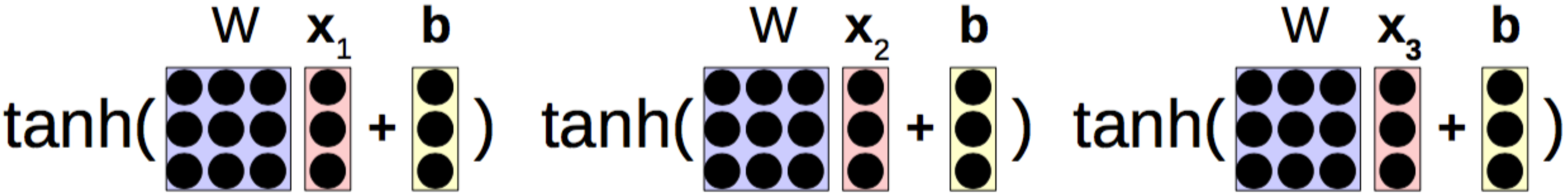
NIPS 2017

Efficiency Tricks: Mini-batching

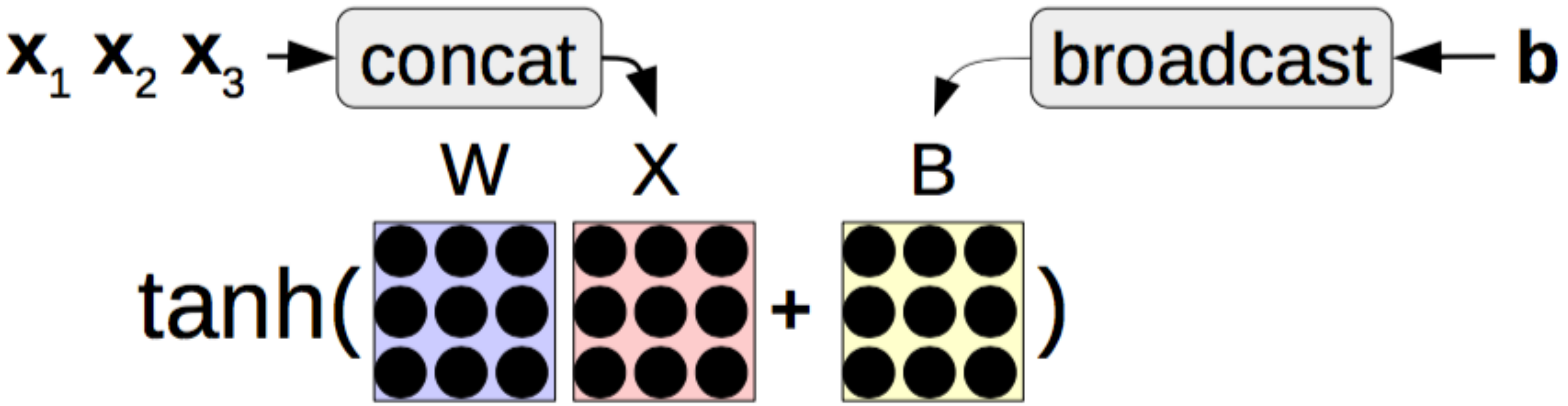
- On modern hardware 10 operations of size 1 is **much slower than** 1 operation of size 10
- Minibatching combines together smaller operations into one big one

Minibatching

Operations w/o Minibatching



Operations with Minibatching



Manual Mini-batching

- In language processing tasks, you need to:
  - Group sentences into a mini batch (optionally, for efficiency group sentences by length)
  - Select the “t”th word in each sentence, and send them to the lookup and loss functions

The Dynamic Neural Network Toolkit

http://dynet.io

- Dynamic graph toolkit implemented in C++, **usable from C++, Python, Scala/Java**
- **Very fast on CPU** (good for prototyping NLP apps!), similar support to other toolkits for GPU
- Support for **on-the-fly batching, implementation of mini-batching**, even in difficult situations

Mini-batched Code Example

```
# in_words is a tuple (word_1, word_2)
# out_label is an output label
word_1 = E[in_words[0]]
word_2 = E[in_words[1]]
scores_sym = W*dy.concatenate([word_1, word_2])+b
loss_sym = dy.pickneglogsoftmax(scores_sym, out_label)
```

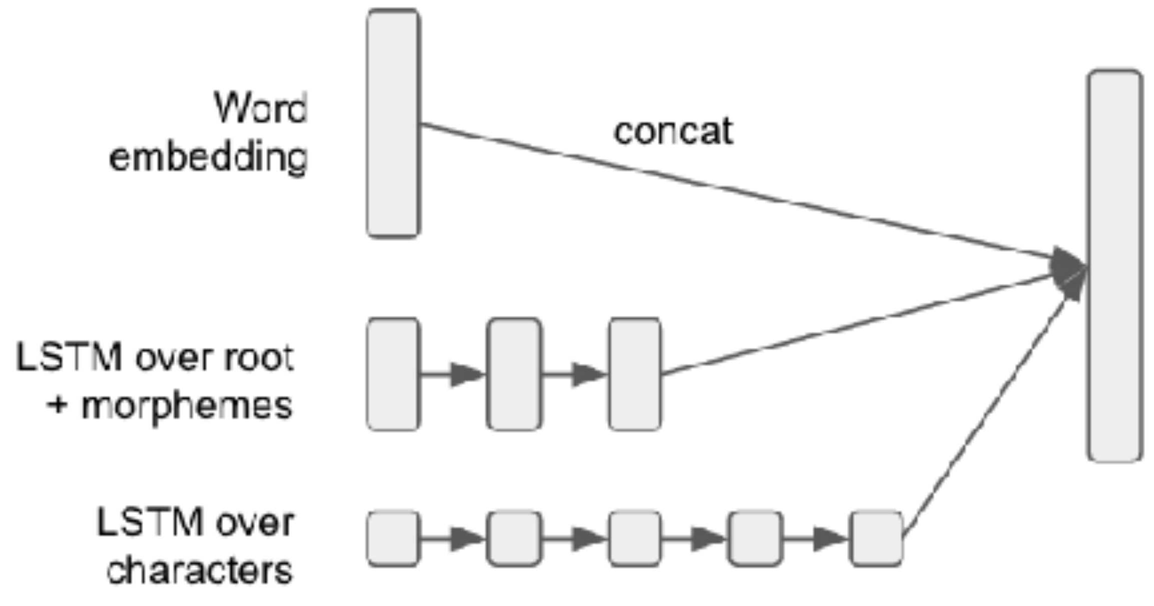
(a) Non-minibatched classification.

```
# in_words is a list [(word_{1,1}, word_{1,2}), (word_{2,1}, word_{2,2}), ...]
# out_labels is a list of output labels [label_1, label_2, ...]
word_1_batch = dy.lookup_batch(E, [x[0] for x in in_words])
word_2_batch = dy.lookup_batch(E, [x[1] for x in in_words])
scores_sym = W*dy.concatenate([word_1_batch, word_2_batch])+b
loss_sym = dy.sum_batches( dy.pickneglogsoftmax_batch(scores_sym, out_labels) )
```

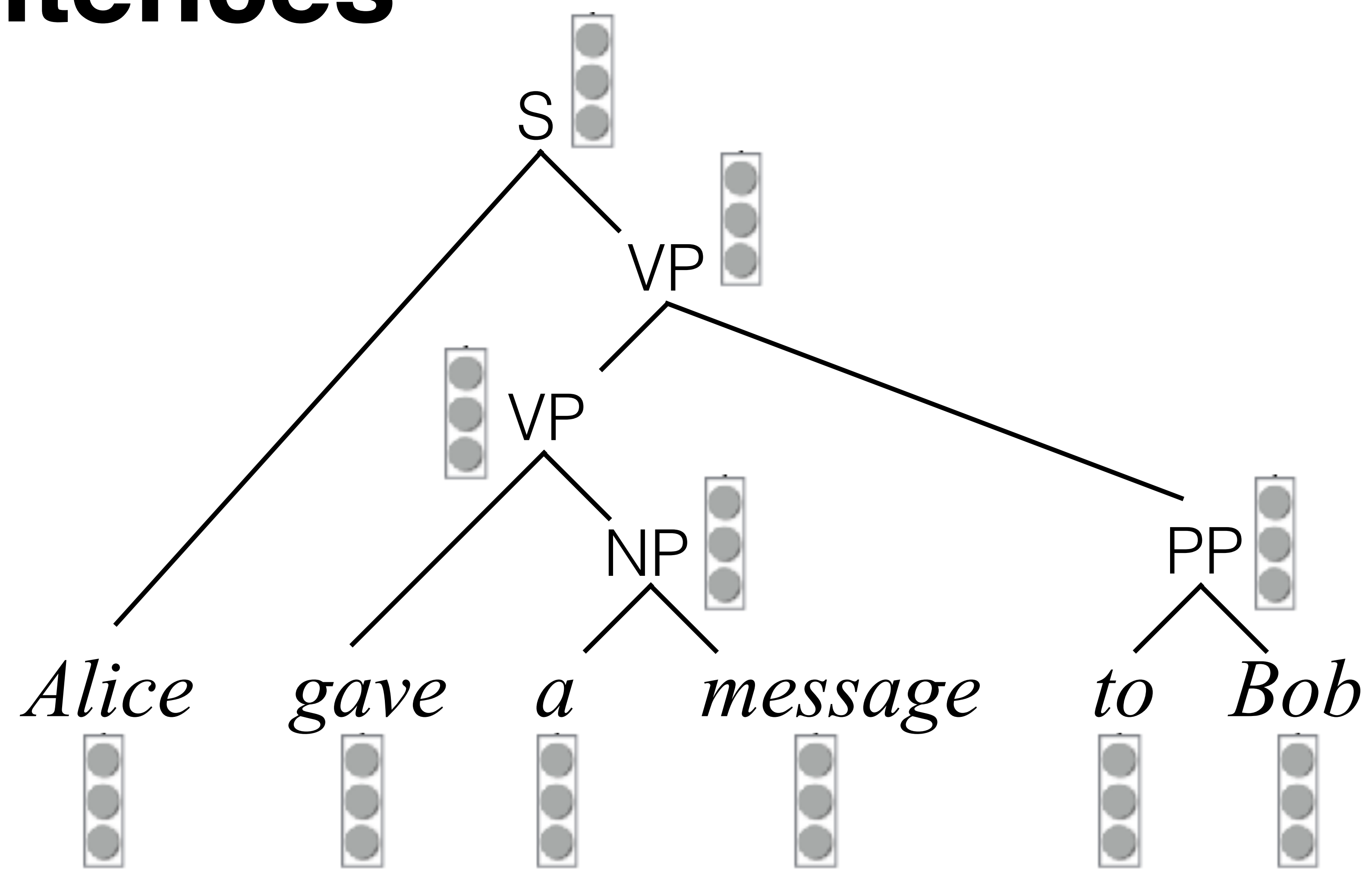
(b) Minibatched classification.

But What about These?

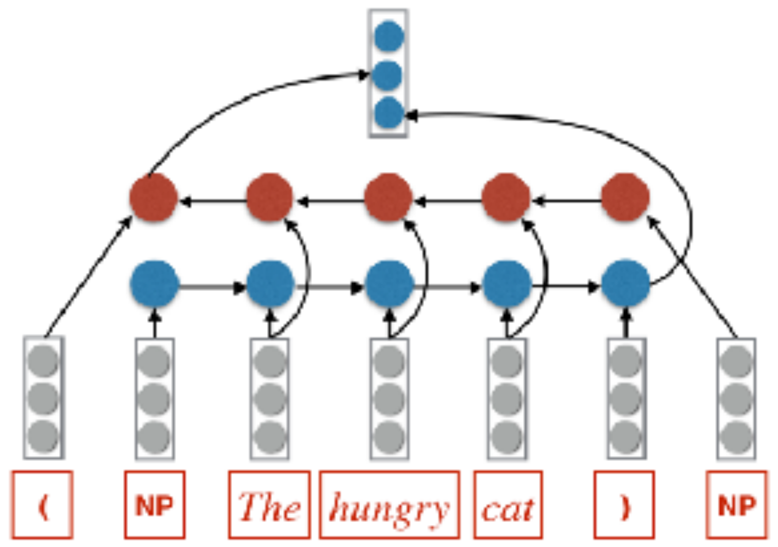
Words



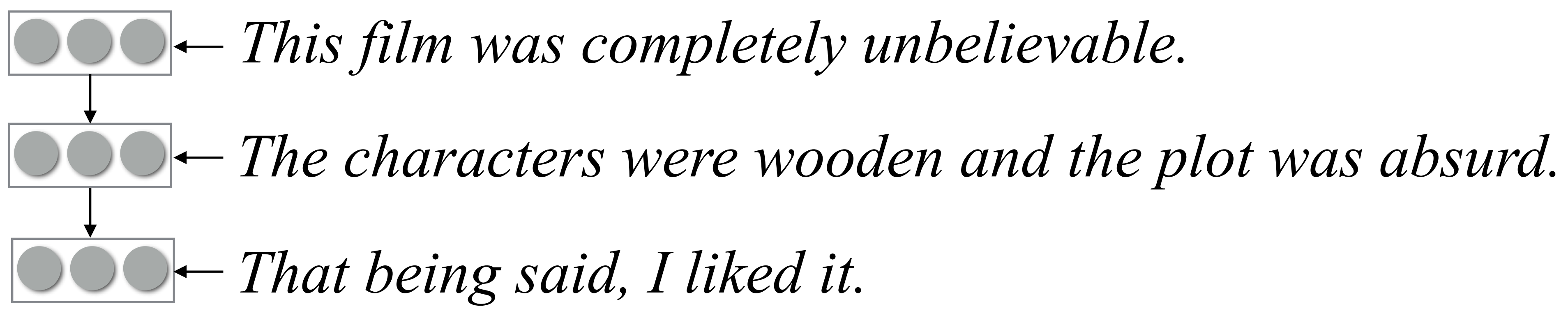
Sentences



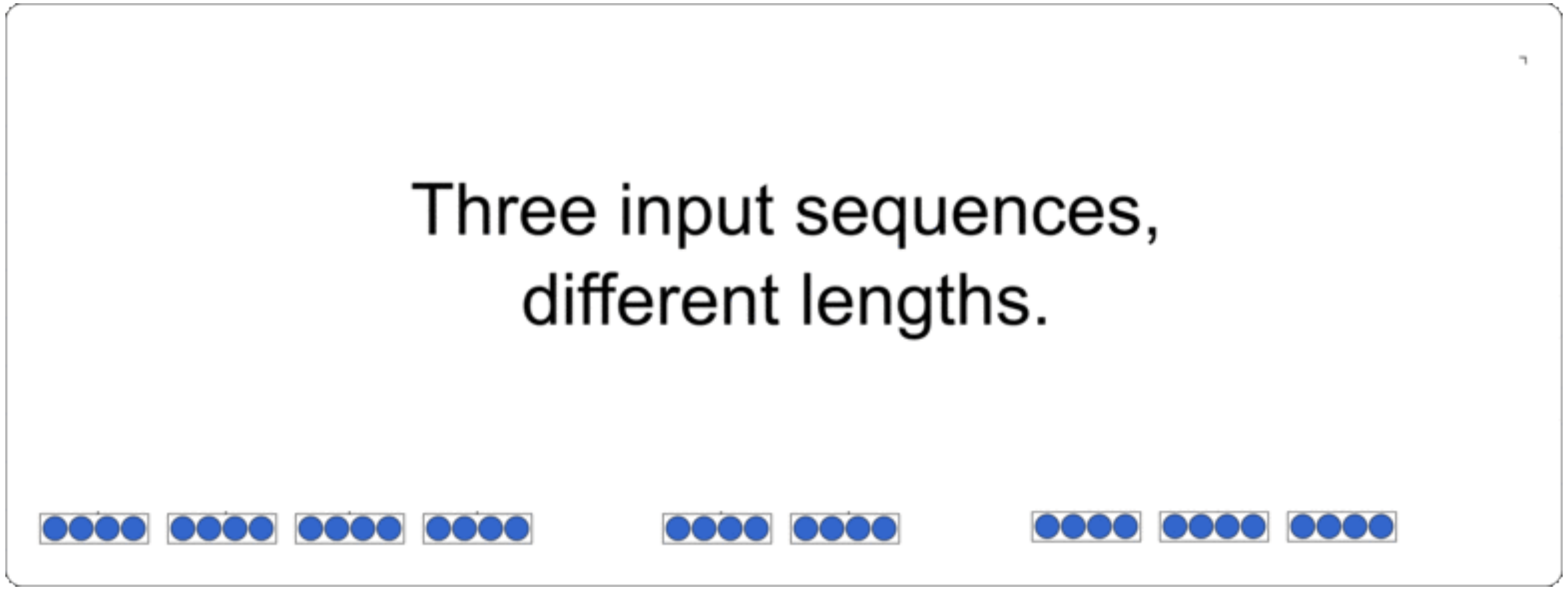
Phrases



Documents



Automatic Mini-batching!

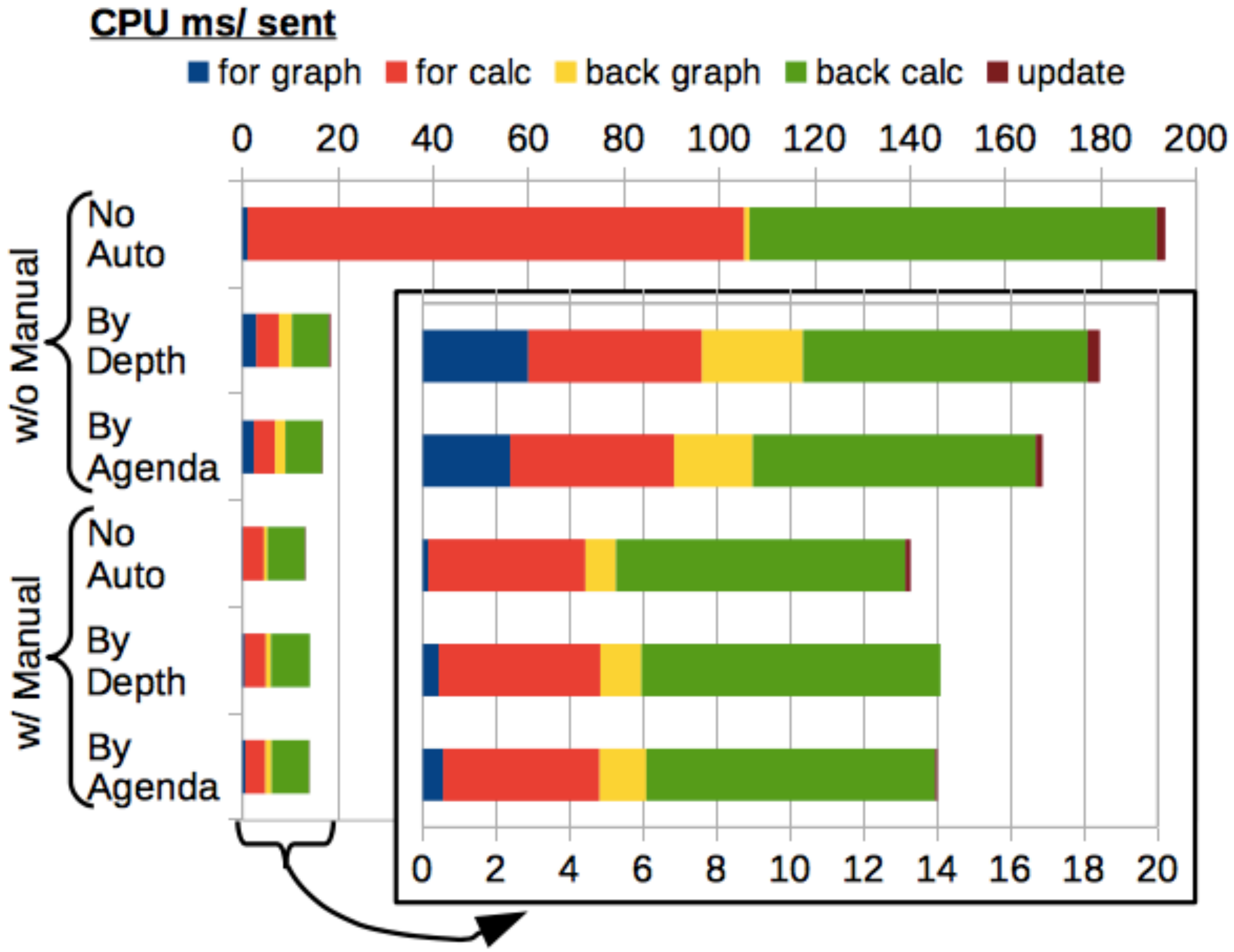


- TensorFlow Fold (complicated combinators)
- DyNet Autobatch (basically effortless implementation)

Autobatching Algorithm

- for each minibatch:
  - for each data point in mini-batch:
    - **define/add data**
  - **sum losses**
  - **forward** (autobatch engine does magic!)
  - **backward**
  - **update**

Speed Improvements



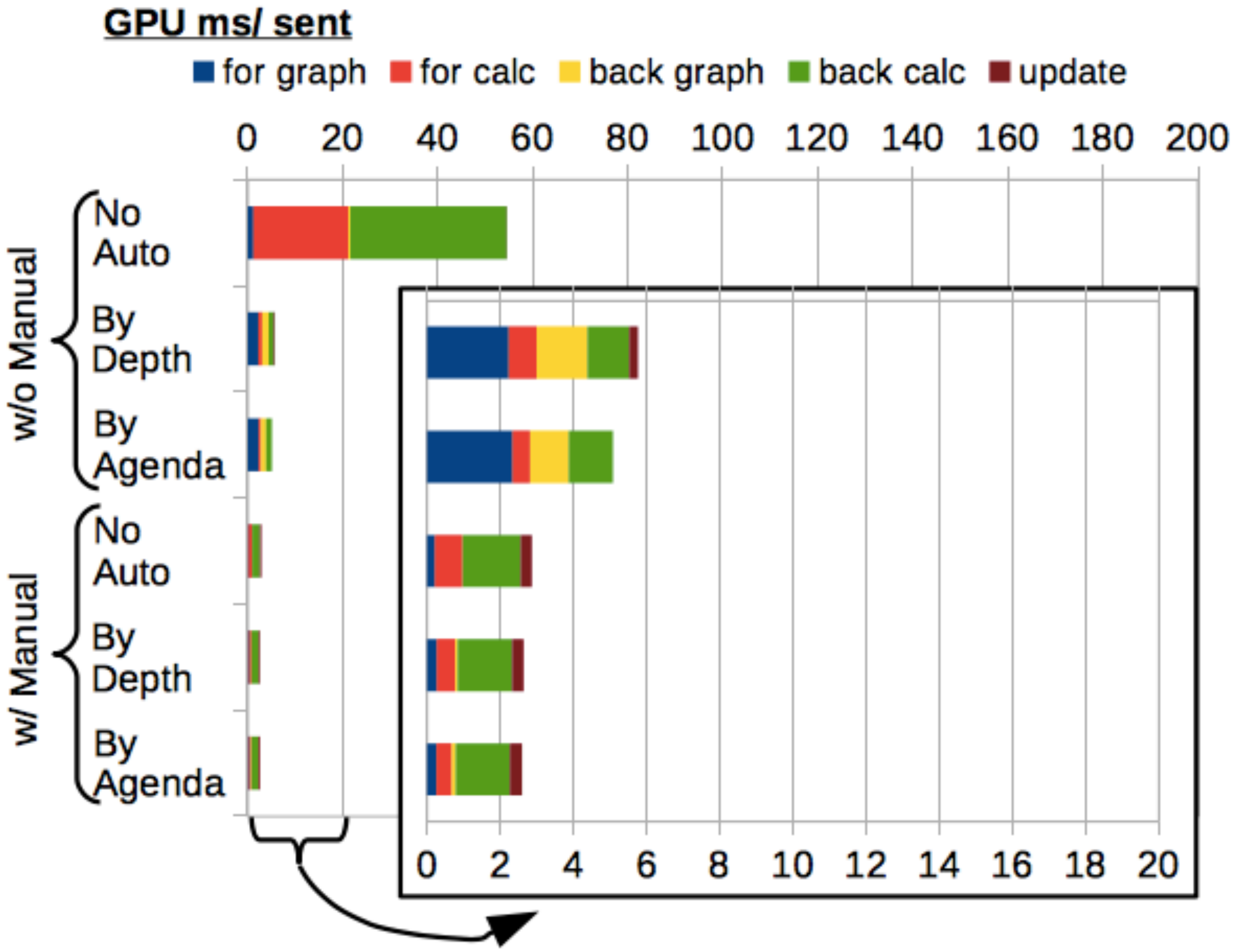


Table 1: Sentences/second on various training tasks for increasingly challenging batching scenarios.

Task	CPU			GPU		
	NoAuto	ByDepth	ByAgenda	NoAuto	ByDepth	ByAgenda
BiLSTM	16.8	139	**156**	56.2	337	**367**
BiLSTM w/ char	15.7	93.8	**132**	43.2	183	**275**
TreeLSTM	50.2	348	**357**	76.5	**672**	661
Transition-Parsing	16.8	61.0	**61.2**	33.0	89.5	**90.1**

Conclusion

Neural Networks as Science

- We all know that neural networks are great for engineering; accuracy gains are undeniable
- But can we also use them as our partners in science?
- Design a net, ask it questions, and see if it’s answers surprise you!

Questions?