Reconciling Vectors and Proofs

Richard Moot
(CNRS, LIRMM)

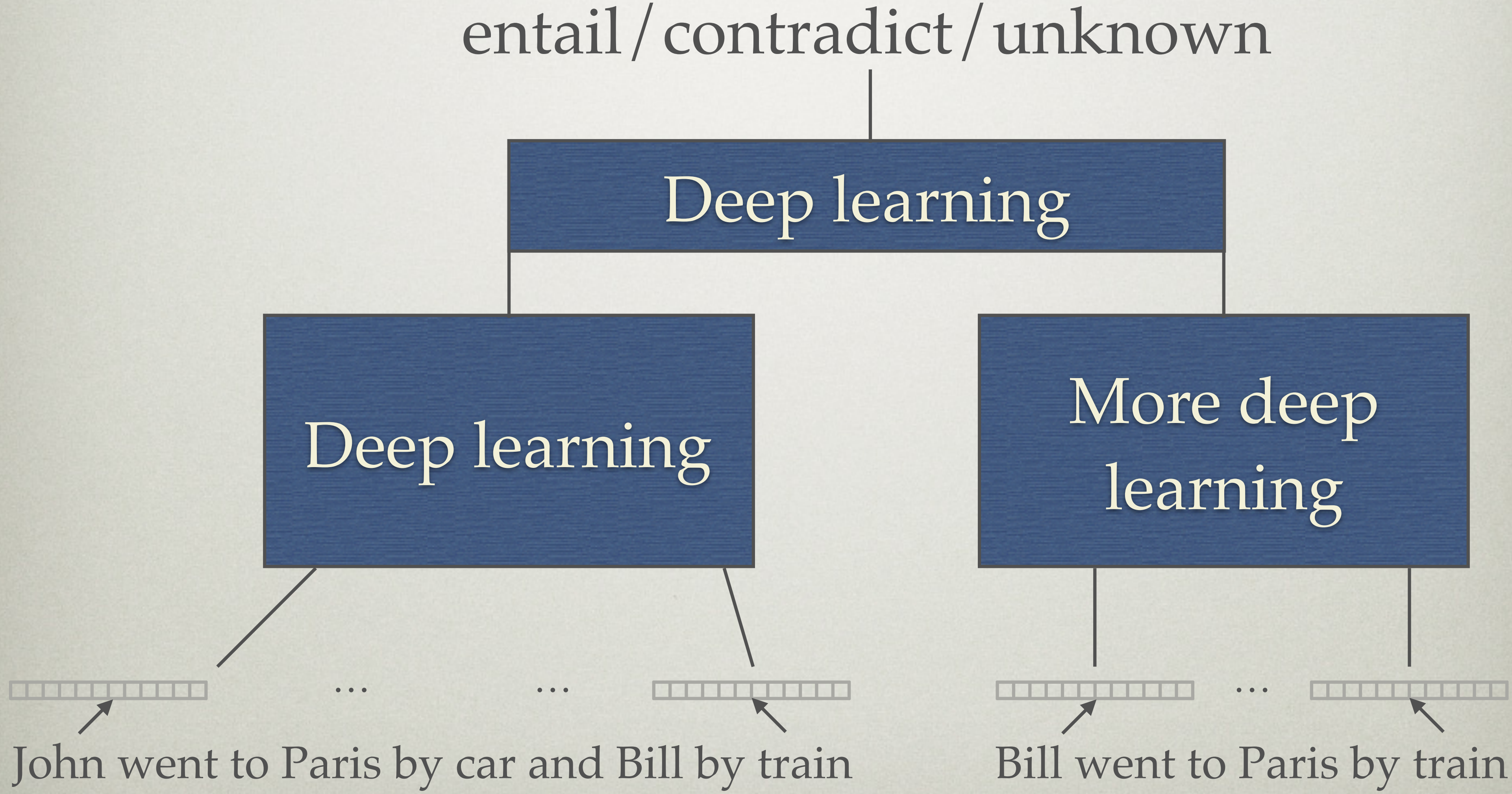
SUPERGROVER SUPERTagging using Grail Over VEctor Representations

Richard Moot
(CNRS, LIRMM)

A Tale of Two Theories of Meaning

- Translate words to vectors
- Translate words to formulas

Vector Based Entailment



Logic Based Entailment

Theorem prover

$F1 \vdash F2$ $F1, F2 \vdash \perp$

F1 = ∃e. went_to_by(e,John,Paris) & by(e,car) &
∃f. went_to(f,Bill,Paris) & by(f,train)

F2 = ∃d.went_to(d,Bill,Paris) & by(d,

TLG theorem prover

np (np\s)/pp pp/np
John went to Paris by car and Bill by train

TLG theorem prover

np (np\s)/pp pp/np
Bill went to Paris by train

Entailment example (from FraCas)

T

John spoke to Mary on Monday.
Bill didn't.

H

Bill didn't speak to Mary on Monday.

Entailment example (from FraCas)

T John went to Paris by car and Bill by train.

H Bill went to Paris by train.

Entailment example (From RTE)

T Eating lots of foods that are a good source of fiber may keep your blood glucose from rising fast after you eat.

H Fiber improves blood sugar control.

Question Answering example (From RACE)

T "Here's a letter for Miss Alice Brown," said the mailman.
" I'm Alice Brown," a girl of about 18 said in a low voice.
Alice looked at the envelope for a minute, and then handed it back to the mailman.
"I'm sorry I can't take it, I don't have enough money to pay it", she said.

Q The girl handed the letter back to the mailman because

A1 she didn't know whose letter it was

A2 she had no money to pay the postage

A3 she received the letter but she didn't want to open it

A4 she had already known what was written in the letter

Question Answering example (From SQuAD)

T In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.

Q What causes precipitation to fall?

A Gravity

Is there Still a Place for Logic?

- There have been enormous advances on the state-of-the art for many hard natural language understanding tasks (XLnet: 86.3 RTE, 98.6 QNLI)
- Is there still a place for logic?

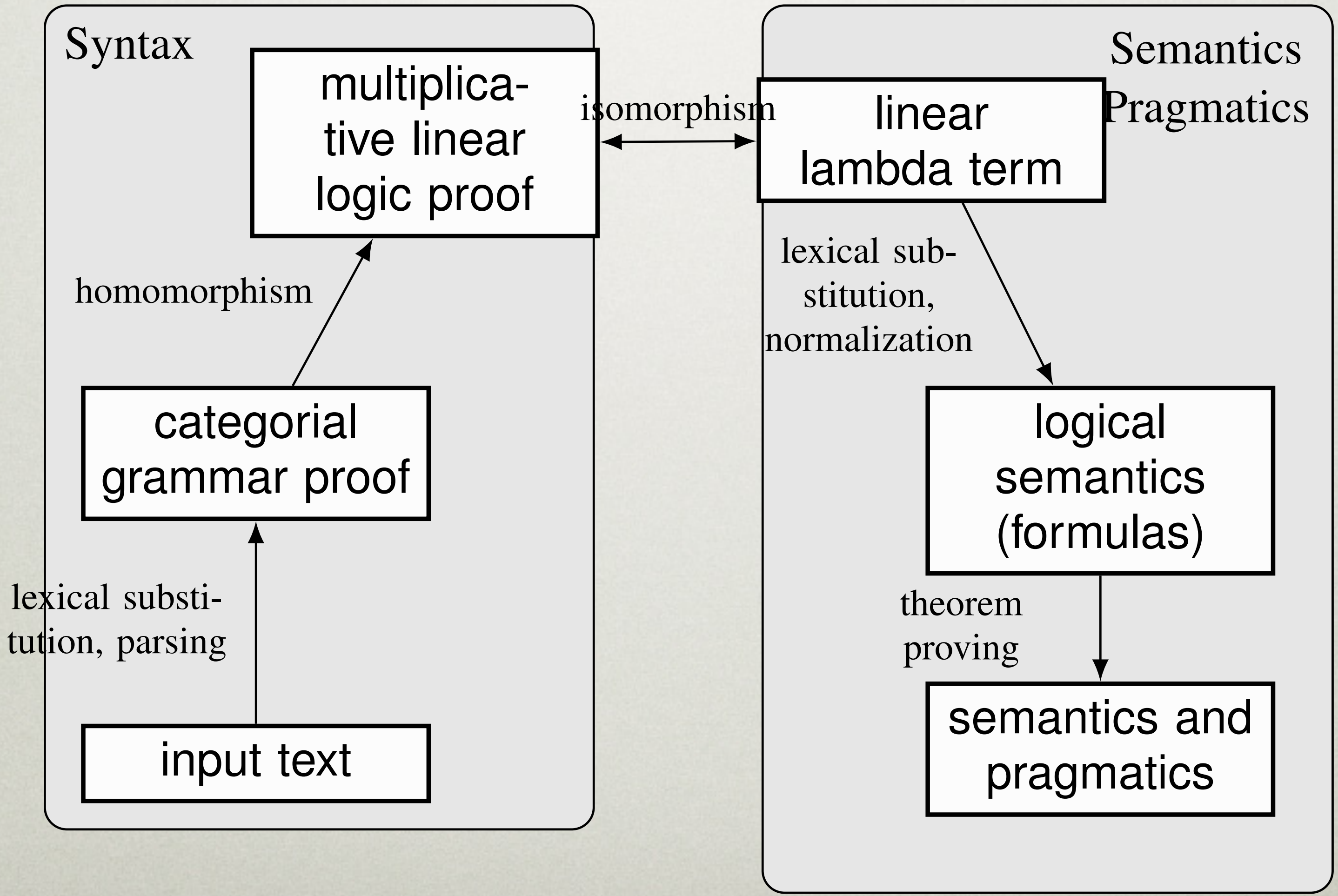
Is there Still a Place for Logic?

- Human annotators are notoriously bad at logical inferences.
- Ideally, we want an inference system which does *logic* at the level of our best theorem provers and *common sense reasoning* at the level of humans.

When To Use Machine Learning?

- We don’t understand what’s going on (eg. describing what is on a picture)
- We do understand, but there is no feasible algorithm (eg. chess, go)

Type-Logical Grammar



Categorial Grammars

Formulas and corresponding expressions

• np	• Jean, l'étudiant, …
• n	• étudiant, économie, …
• s	• Jean dort, Jean aime Marie
• np\s	• dort, aime Marie
• np/n	• un, chaque, l'
• (np\s)/np	• aime, étudie

Categorial Grammars

Rules

Lambek categorial grammars have only four rules: an elimination and an introduction rule for both "\" and "/"

$$\frac{A/B \qquad B}{A}[/E] \qquad \frac{B \qquad B\backslash A}{A}[\backslash E]$$

$$\frac{\begin{array}{cc} \ldots & [B]^i \\ \vdots & \\ A & \end{array}}{A/B}[/I]^i \qquad \frac{\begin{array}{cc} [B]^i & \ldots \\ & \vdots \\ & A \end{array}}{B\backslash A}[\backslash I]^i$$

Categorial Grammars

Example

un	étudiant	dort
np/n	n	$np\backslash s$

$$\frac{A/B \qquad B}{A}[/E] \qquad \frac{B \qquad B\backslash A}{A}[\backslash E]$$

$$\frac{\begin{array}{c}\ldots \qquad [B]^i \\ \vdots \\ A\end{array}}{A/B}[/I]^i \qquad \frac{\begin{array}{c}[B]^i \qquad \ldots \\ \vdots \\ A\end{array}}{B\backslash A}[\backslash I]^i$$

Categorial Grammars

Example

$$
\begin{array}{ccc}
\text{un} & \text{étudiant} & \text{dort} \\
\multicolumn{2}{c}{\dfrac{np/n \qquad n}{np}\,[/E]} & np\backslash s
\end{array}
$$

$$
\dfrac{A/B \qquad B}{A}\,[/E] \qquad \dfrac{B \qquad B\backslash A}{A}\,[\backslash E]
$$

$$
\dfrac{\begin{array}{c} \ldots \quad [B]^i \\ \vdots \\ A \end{array}}{A/B}\,[/I]^i \qquad \dfrac{\begin{array}{c} [B]^i \quad \ldots \\ \vdots \\ A \end{array}}{B\backslash A}\,[\backslash I]^i
$$

Categorial Grammars

Example

$$\cfrac{\cfrac{\overset{\text{un}}{np/n} \qquad \overset{\text{étudiant}}{n}}{np}\,[/E] \qquad \overset{\text{dort}}{np\backslash s}}{s}\,[/E]$$

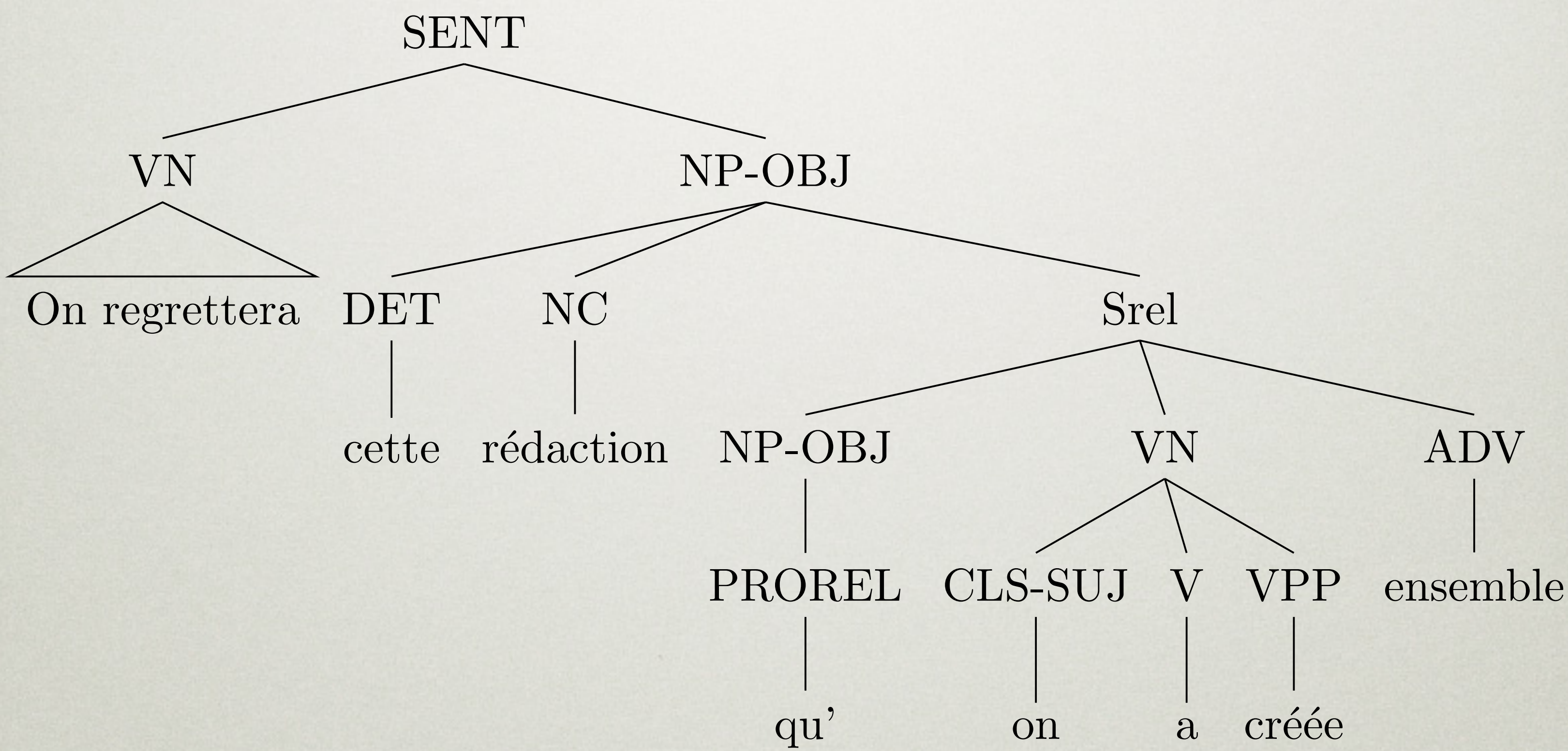
$$\frac{A/B \qquad B}{A}\,[/E] \qquad \frac{B \qquad B\backslash A}{A}\,[\backslash E]$$

$$\frac{\begin{array}{c}\ldots \qquad [B]^i \\ \vdots \\ A\end{array}}{A/B}\,[/I]^i \qquad \frac{\begin{array}{c}[B]^i \qquad \ldots \\ \vdots \\ A\end{array}}{B\backslash A}\,[\backslash I]^i$$

Lambek grammars and beyond

- getting the semantics right requires a somewhat richer system than AB grammars
- introduction rules (“traces” or the original “slash categories” and their semantics)
- structural rules (“movement” or “head wrap”, essentially restricted tree rewrite operations)

Introduction Rules: Example



Introduction Rules: Example

redaction	qu'	on	a	créée
n	$(n\backslash n)/(s/np)$	np	$(np\backslash s)/(np\backslash s_{ppart})$	$(np\backslash s_{ppart})/np$

Introduction Rules: Example

redaction	qu'	on	a	créée	
n	$(n\backslash n)/(s/np)$	np	$(np\backslash s)/(np\backslash s_{ppart})$	$(np\backslash s_{ppart})/np$	np

Introduction Rules: Example

$$
\begin{array}{cccccc}
\text{redaction} & \text{qu'} & \text{on} & \text{a} & \text{créée} & \\
n & (n\backslash n)/(s/np) & np & (np\backslash s)/(np\backslash s_{ppart}) & \dfrac{(np\backslash s_{ppart})/np \qquad np}{np\backslash s_{ppart}}\,[/E] &
\end{array}
$$

Introduction Rules: Example

redaction qu' on a créée

$$
n \quad (n\backslash n)/(s/np) \quad np \quad \dfrac{(np\backslash s)/(np\backslash s_{ppart}) \qquad \dfrac{(np\backslash s_{ppart})/np \qquad np}{np\backslash s_{ppart}}[/E]}{np\backslash s}[/E]
$$

Introduction Rules: Example

redaction qu' on a créée

$$\begin{array}{ccc} \text{redaction} & \text{qu'} & \\ n & (n\backslash n)/(s/np) & \dfrac{\underset{\text{on}}{np} \qquad \dfrac{\underset{\text{a}}{(np\backslash s)/(np\backslash s_{ppart})} \qquad \dfrac{\underset{\text{créée}}{(np\backslash s_{ppart})/np} \qquad np}{np\backslash s_{ppart}}\,[/E]}{np\backslash s}\,[/E]}{s}\,[\backslash E] \end{array}$$

Introduction Rules: Example

$$
\begin{array}{cc}
\text{redaction} & \text{qu'} \\
n & (n\backslash n)/(s/np)
\end{array}
\qquad
\dfrac{\dfrac{\overset{\text{on}}{np} \qquad \dfrac{\overset{\text{a}}{(np\backslash s)/(np\backslash s_{ppart})} \qquad \dfrac{\overset{\text{créée}}{(np\backslash s_{ppart})/np} \qquad [np]^1}{np\backslash s_{ppart}}\,[/E]}{np\backslash s}\,[/E]}{s}\,[\backslash E]}{s/np}\,[/I]^1
$$

Introduction Rules: Example

$$
\overset{\text{redaction}}{n} \qquad
\dfrac{\overset{\text{qu'}}{(n\backslash n)/(s/np)} \quad
\dfrac{\dfrac{\overset{\text{on}}{np} \quad
\dfrac{\overset{\text{a}}{(np\backslash s)/(np\backslash s_{ppart})} \quad
\dfrac{\overset{\text{créée}}{(np\backslash s_{ppart})/np} \quad [np]^1}{np\backslash s_{ppart}}\,[/E]}{np\backslash s}\,[/E]}{s}\,[\backslash E]}{s/np}\,[/I]^1}{n\backslash n}\,[/E]
$$

Introduction Rules: Example

$$\dfrac{\begin{array}{c}\text{redaction}\\ n\end{array}\qquad \dfrac{\begin{array}{c}\text{qu'}\\ (n\backslash n)/(s/np)\end{array}\qquad \dfrac{\dfrac{\begin{array}{c}\text{on}\\ np\end{array}\qquad \dfrac{\begin{array}{c}\text{a}\\ (np\backslash s)/(np\backslash s_{ppart})\end{array}\qquad \dfrac{\begin{array}{c}\text{créée}\\ (np\backslash s_{ppart})/np\end{array}\qquad [np]^1}{np\backslash s_{ppart}}[/E]}{np\backslash s}[/E]}{s}[\backslash E]}{s/np}[/I]^1}{n\backslash n}[/E]}{n}$$

Lambda Calculus and Proofs as terms

- Proofs in categorial grammar correspond to lambda terms
- These lambda terms "forget" the directions of the implications.

$$\frac{t{:}A/B \qquad u{:}B}{(t\ u){:}A} \qquad \frac{u{:}B \qquad t{:}B\backslash A}{(t\ u){:}A}$$

$$\frac{\begin{array}{c}[x{:}B] \\ \vdots \\ t{:}A\end{array}}{A/B{:}\lambda x.t} \qquad \frac{\begin{array}{c}[x{:}B] \\ \vdots \\ t{:}A\end{array}}{B\backslash A{:}\lambda x.t}$$

Why Parsing is Important

Why Parsing is Important

- Killer sentenced to die for second time in 10 years.
- Enraged cow injures farmer with axe
- Top stories: … Obama-Castro handshake and same-sex marriage date set

Why Parsing is Important

- Killer sentenced to die for second time in 10 years.

killer *sentenced_to* *die* *for second…*

$$\mathrm{np} \qquad \frac{(\mathrm{np}\backslash \mathrm{s})/(\mathrm{np}/\mathrm{s}) \quad \mathrm{np}/\mathrm{s}}{\mathrm{np}/\mathrm{s}} \qquad (\mathrm{np}\backslash \mathrm{s})\backslash(\mathrm{np}/\mathrm{s})$$

Why Parsing is Important

- Killer sentenced to die for second time in 10 years.

sentenced_to *die*

$(np\backslash s)/(np/s)$ np/s *for second…*

killer np/s $(np\backslash s)\backslash(np/s)$

np np/s

s

Why Parsing is Important

- Killer sentenced to die for second time in 10 years.

killer	*sentenced_to*	*die*	*for second…*
np	(np\s)/(np/s)	np/s	(np\s)\(np/s)

$$\frac{np/s \quad (np\backslash s)\backslash(np/s)}{np/s}$$

Why Parsing is Important

- Killer sentenced to die for second time in 10 years.

die *for second…*

sentenced_to np/s (np\s)\(np/s)

killer (np\s)/(np/s) np/s

np np/s

s

Why Parsing is Important

- Killer sentenced to die for second time in 10 years.
- Enraged cow injures farmer with axe
- Top stories: … Obama-Castro handshake and same-sex marriage date set

Why Parsing is Important

- Killer sentenced to die for second time in 10 years.
- Enraged cow injures farmer with axe
- Top stories: … Obama-Castro handshake and same-sex marriage date set

Why Parsing is Important

- Killer sentenced to die for second time in 10 years.
- Enraged cow injures farmer with axe
- Top stories: … Obama-Castro handshake and same-sex marriage date set

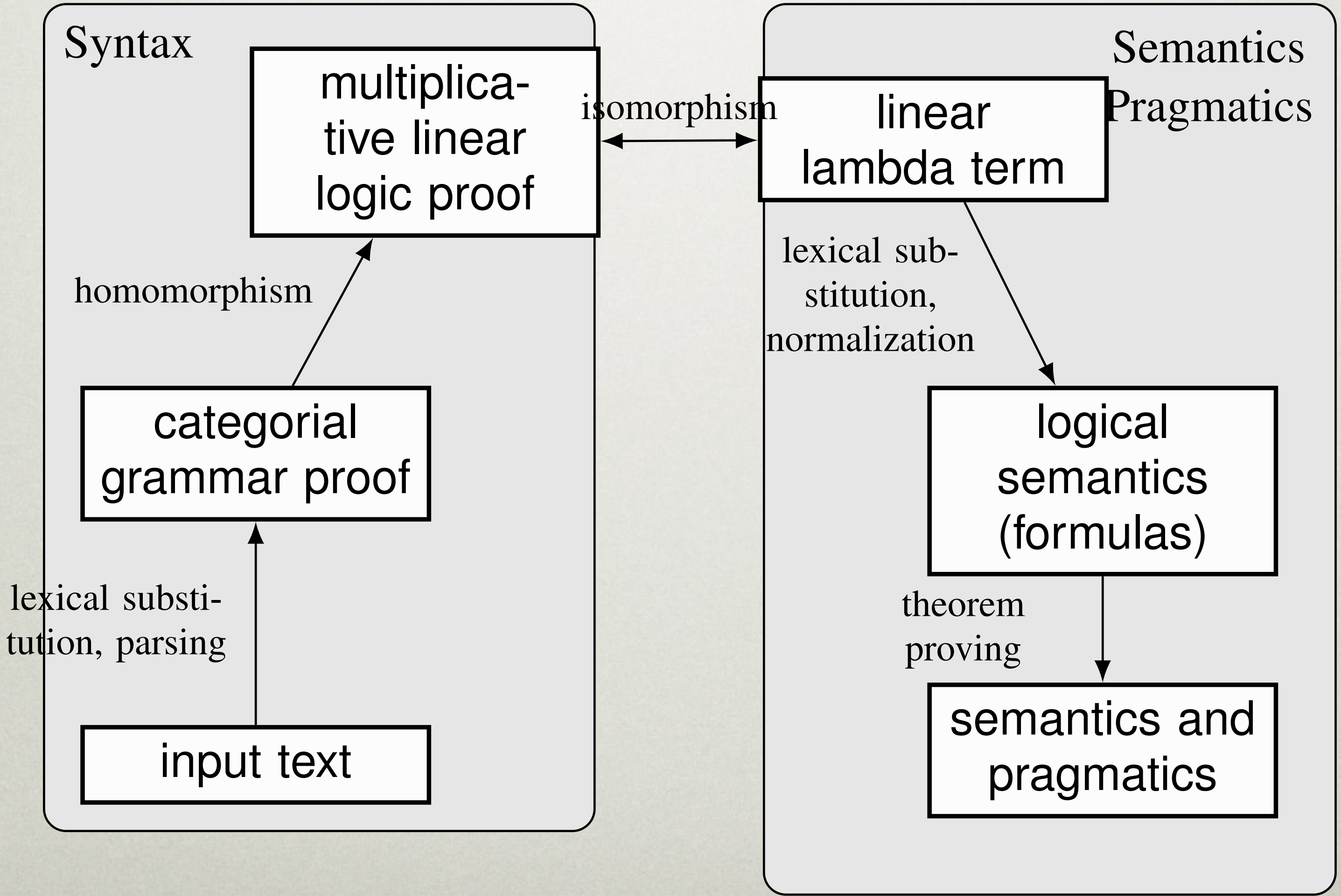
Why Parsing is Important

- Killer sentenced to [die for second time in 10 years].
- Enraged cow [[injures farmer] with axe]
- Top stories: … Obama-Castro [[handshake and same-sex marriage] date set]

Why Parsing is Important

- Killer [sentenced to die] for second time in 10 years.
- Enraged cow injures [farmer with axe]
- Top stories: … [Obama-Castro handshake] and [same-sex marriage date set]

Type-Logical Grammar



Where Could Machine Learning Be Useful?

- Assigning formulas to word (supertagging)
- Choosing the “best” proof among alternatives
- Computing entailments in the target logic (reinforcement learning)

Wide-Coverage Parsing

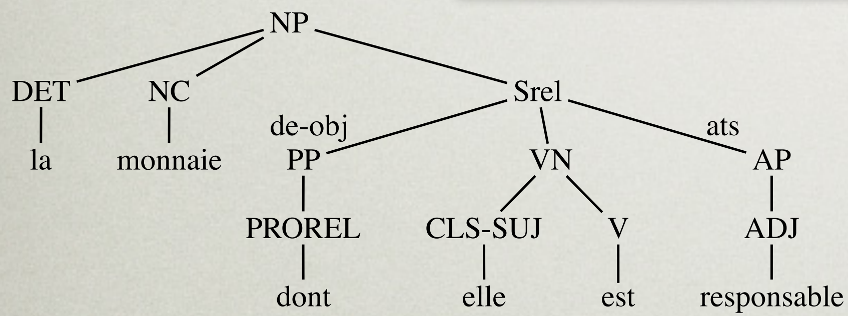
- How can we parse arbitrary text with type-logical grammars?
- Before we can even start, we need a sufficiently large lexicon.
- How can we assign a formula to a word we have never seen with this formula? Maybe we have never seen the word at all.

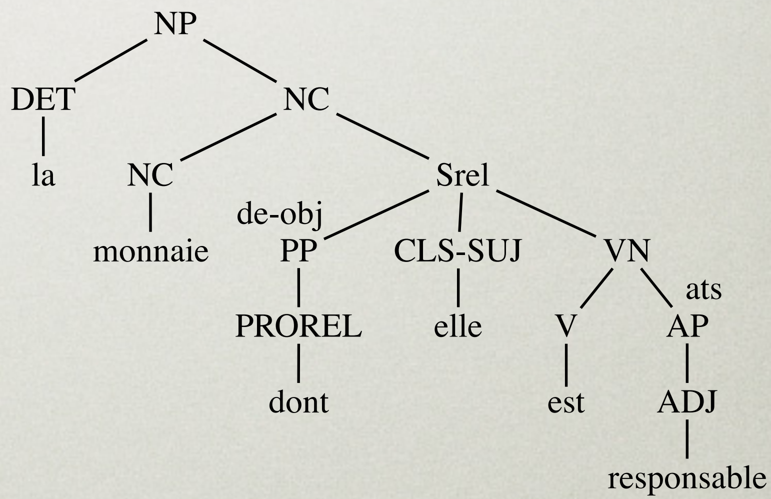
Wide-Coverage Parsing

- No good theoretical model of the "right" formula for the words in a sentence.
- Maybe a case for machine learning?
- However, we need a fair amount of data

Treebank Extraction

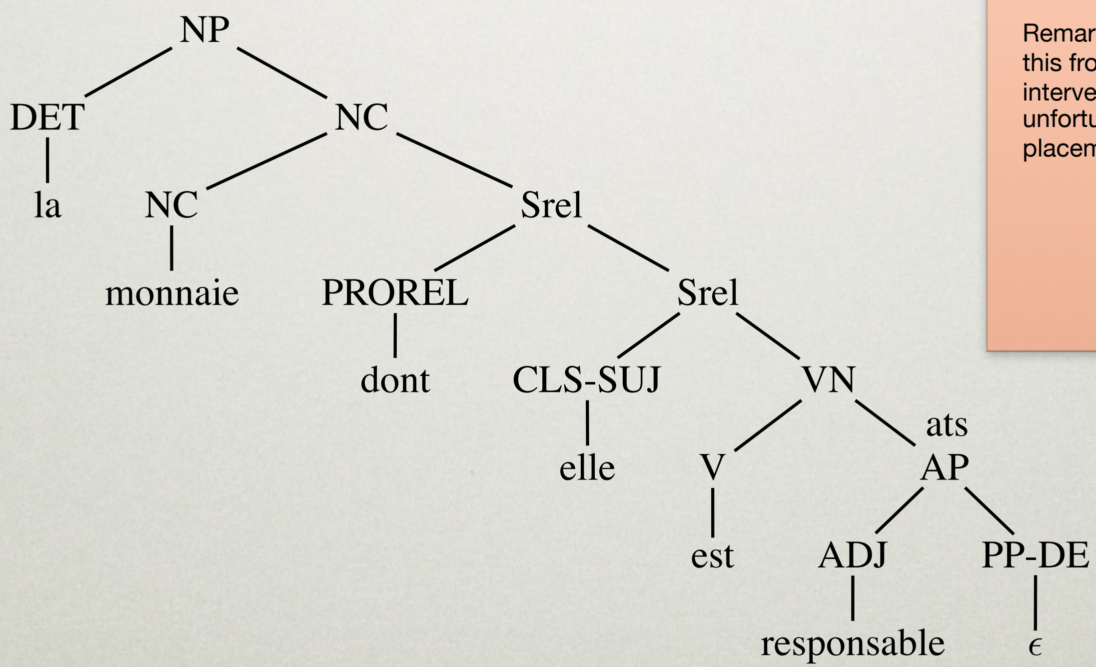
Sentence as we find it in the corpus. “dont” is a relative pronoun like “que” but which selects a sentence missing de “de” preposition (instead of a sentence missing an np like “que”)





Note how “dont” is annotated as a “de-obj” argument, which is useful.

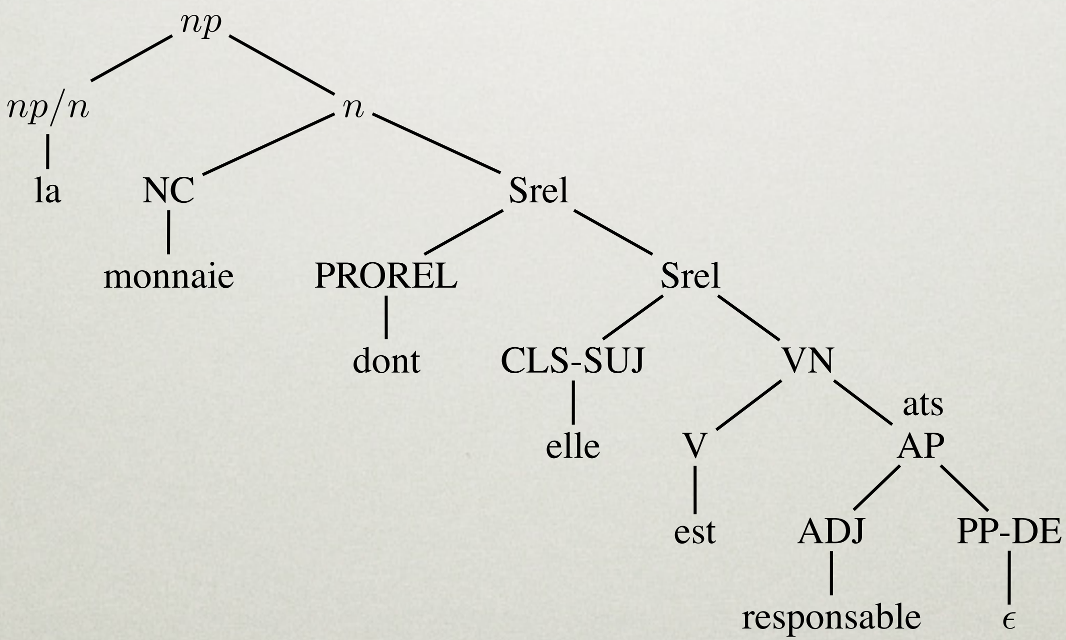
Treebank Extraction



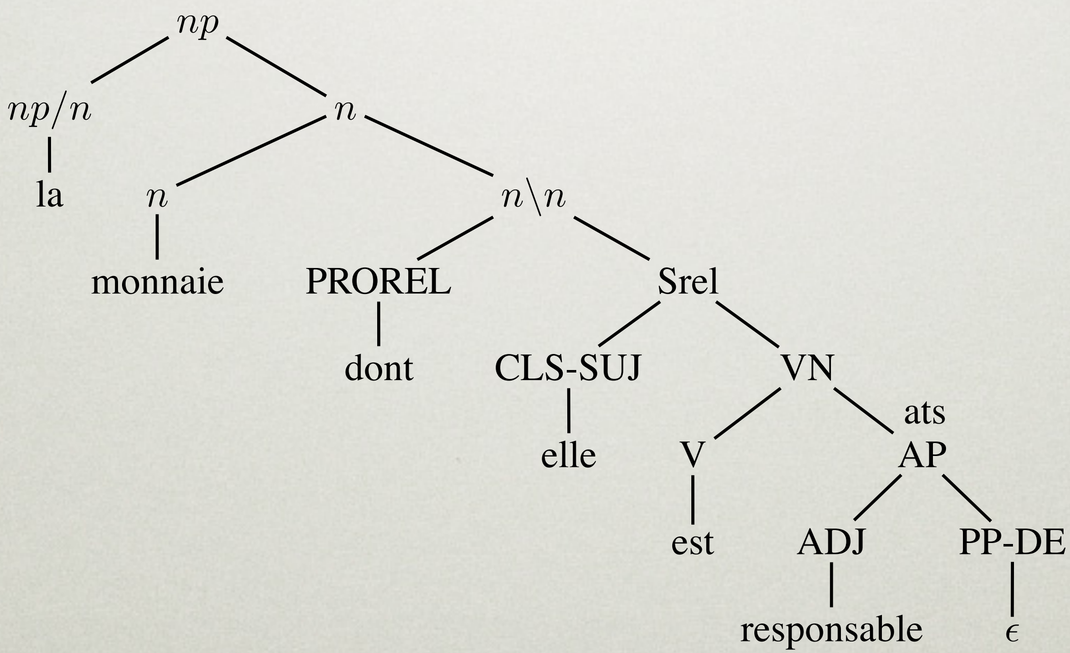
However, the “de” preposition belongs to “responsable” (some adjectives select for prepositions: “responsable de X” functions as an adjective just as “responsable”)

Remark however, that there is no way to derive this from the annotation as it is given. Manual intervention (or at least verification!) is unfortunately necessary to assure the correct placement of the hypothetical preposition.

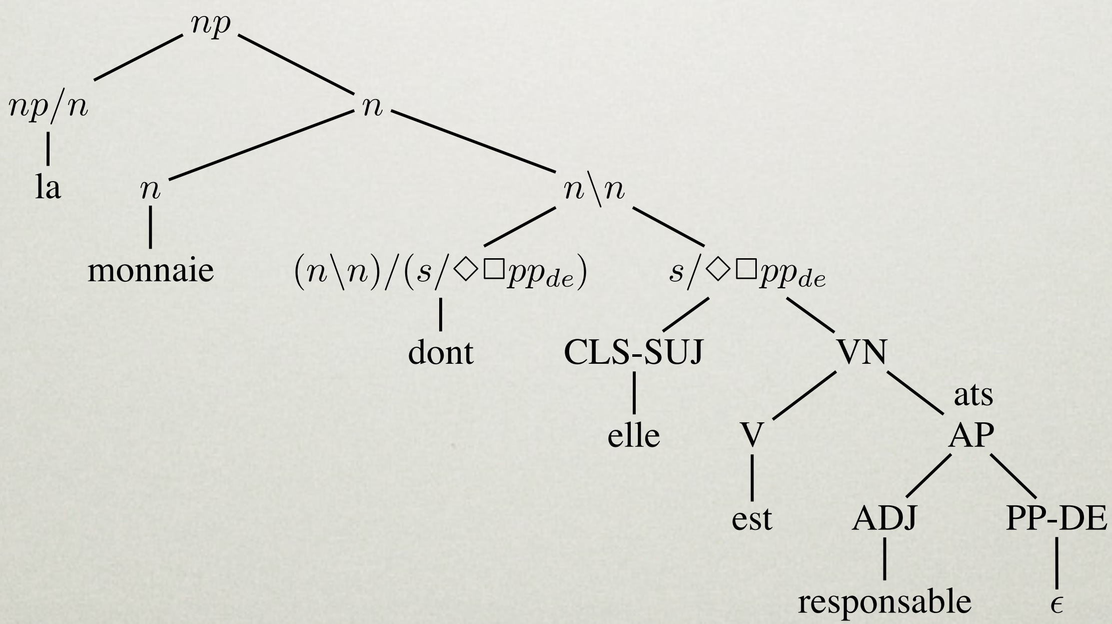
Treebank Extraction



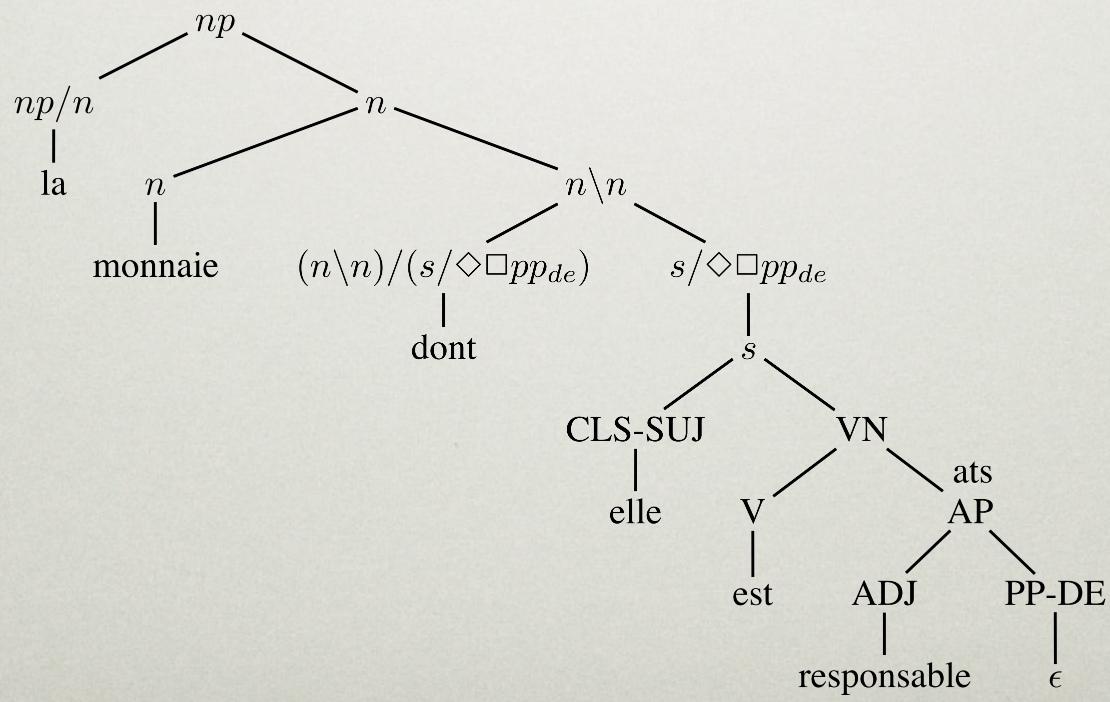
Treebank Extraction



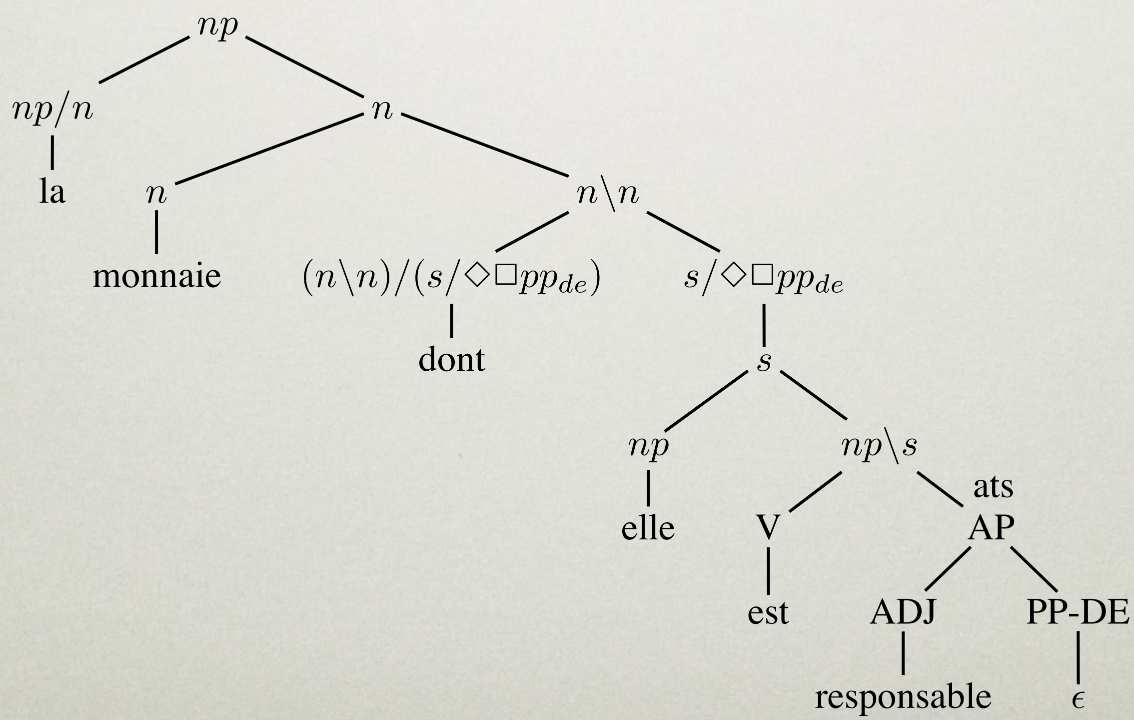
Treebank Extraction



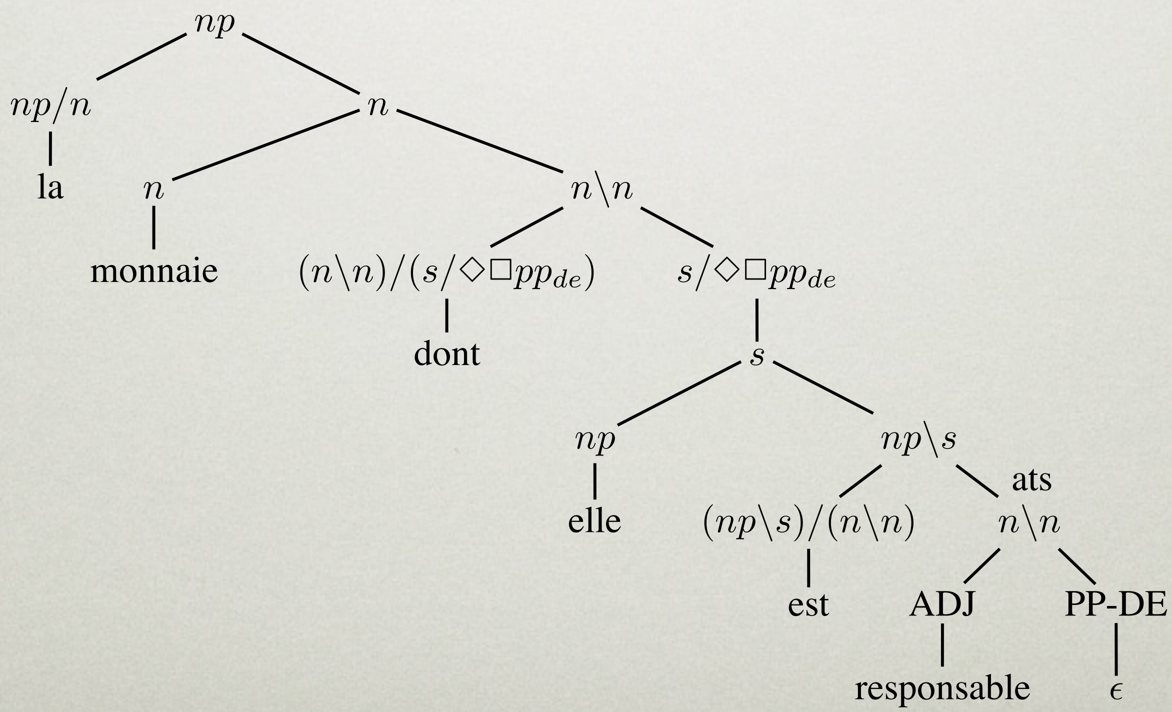
Treebank Extraction



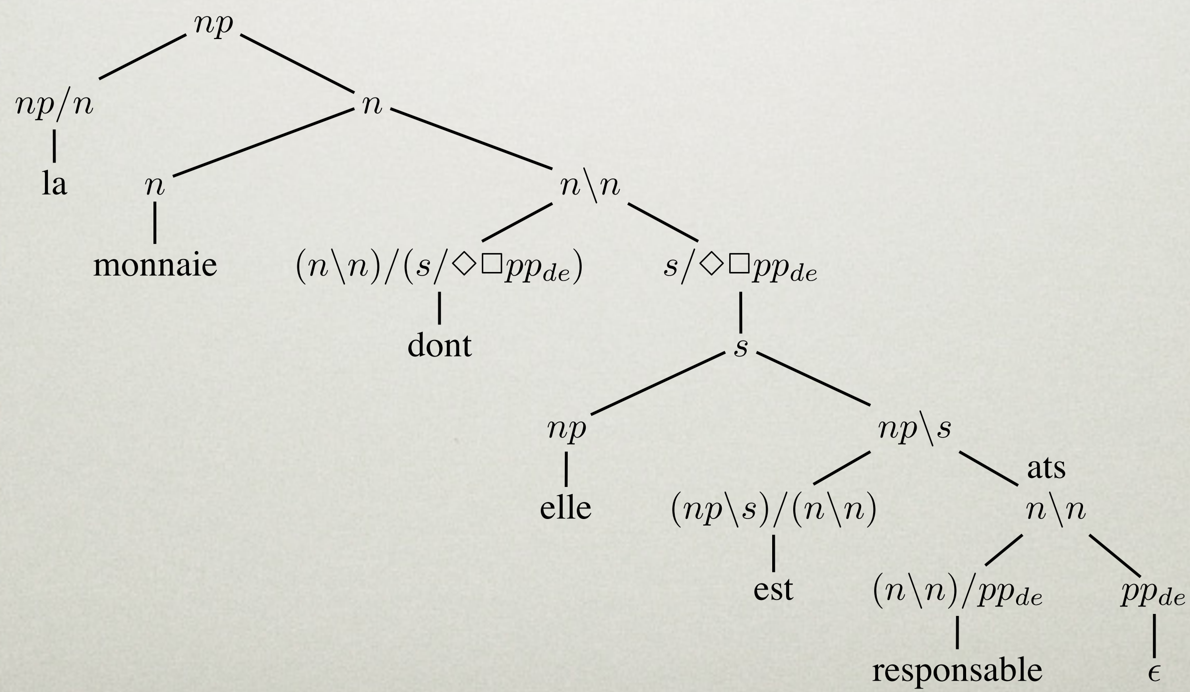
Treebank Extraction



Treebank Extraction



Treebank Extraction



Statistics About the Extracted French Treebank

- 15,590 sentences 445,918 words
- 43,098 distinct lexical entries
- 859 different formulas
- By comparison: 12,617 CFG rules

Note: these are the statistics after considerable cleanup: the first version had over 4,000 different formulas!

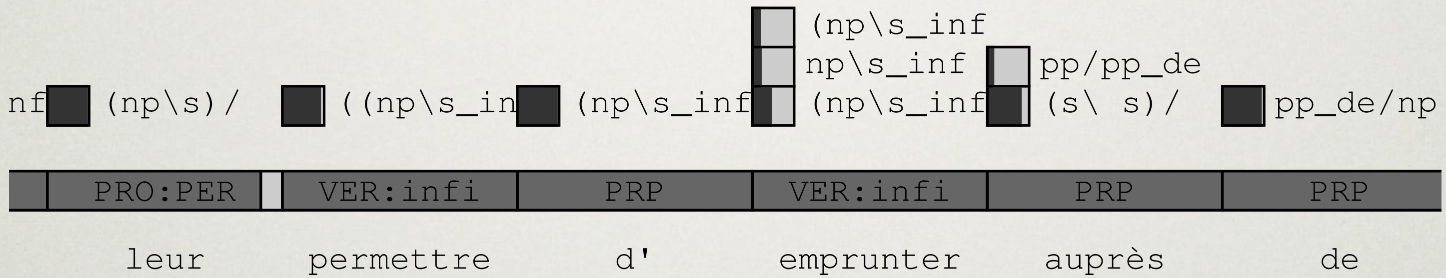
Question: can we actually use the extracted grammar for parsing?

Lexicon Size

- Many frequent words occur with very many different formulas
- Classic solution: supertagging

est - "is"	
$(np\backslash s)/np$	23,2 %
$(np\backslash s)/(n\backslash n)$	20,6 %
$(np\backslash s)/(np\backslash s_{pass})$	16,8 %
$(cl_r\backslash(np\backslash s))/(cl_r\backslash(np\backslash s_{ppart}))$	10,8 %
$(np\backslash s)/pp$	8,1 %
$(np\backslash s)/(np\backslash s_{ppart})$	6,3 %
$(np\backslash s)/(np\backslash s_{infX})$	2,8 %
$((np\backslash s)/s_q)/(n\backslash n)$	2,2 %

What Supertagging Does



- Supertagging ≅ statistical approximation of lexical lookup
- Assigns each word the contextually most likely (set of) formulas

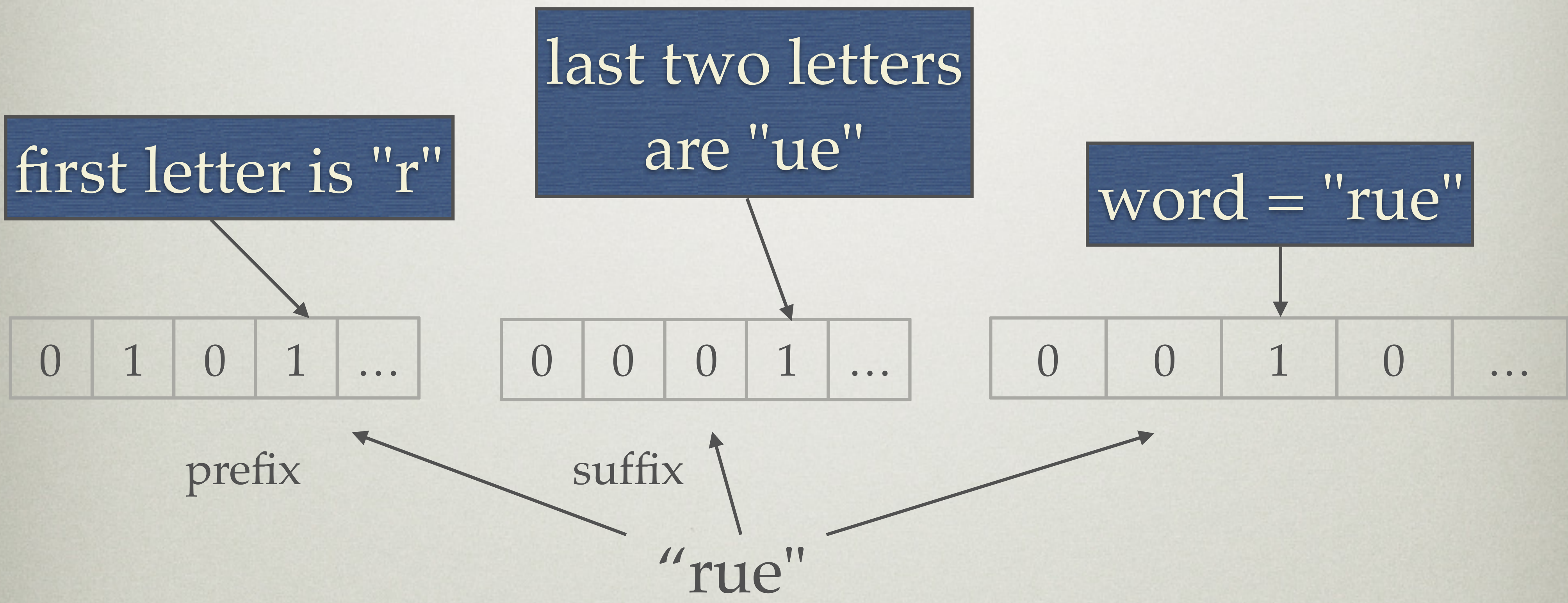
Minimal Fusion

Word to vector then vector to formula
But which vectors?

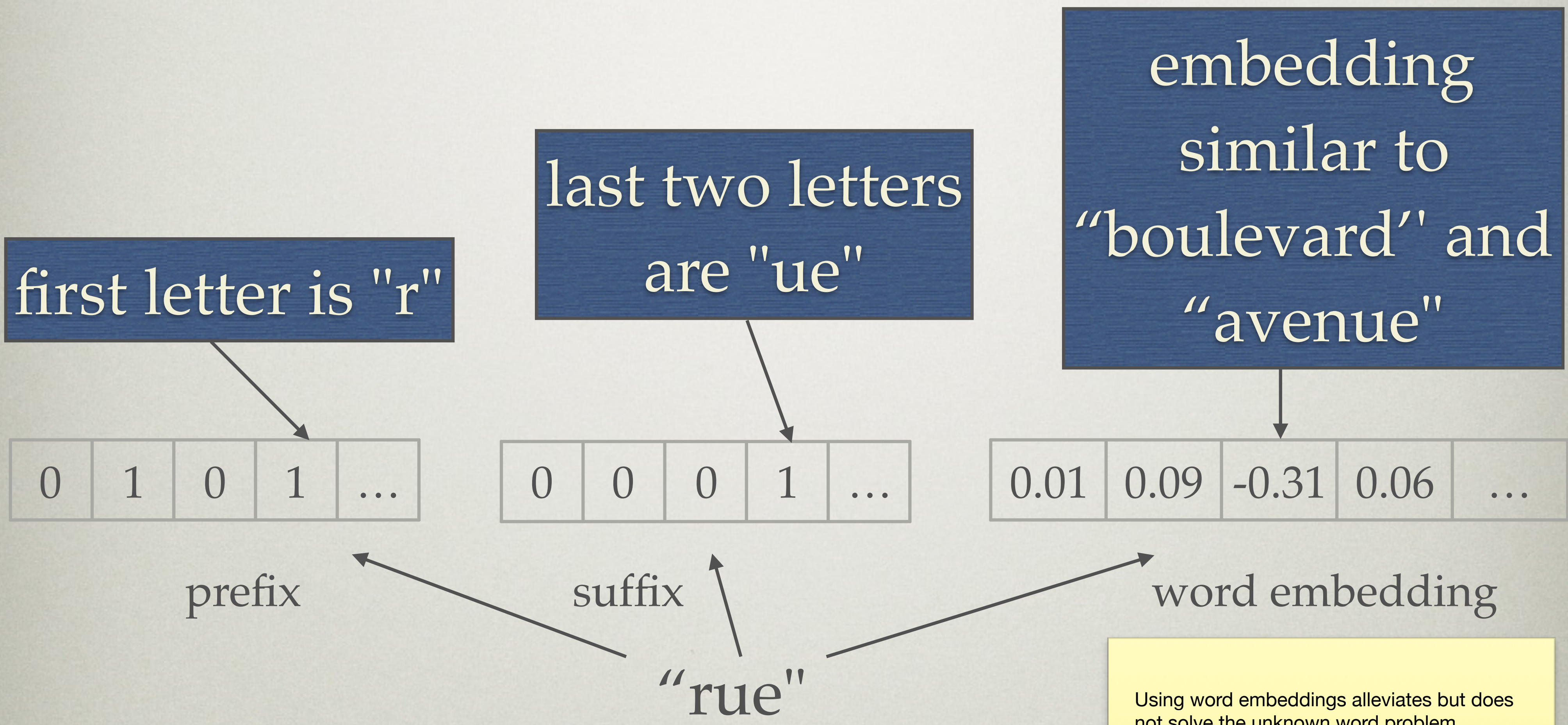
What do we use as inputs to our models?

- represent each word by a fixed-length vector
- vector representation must contain enough information for downstream tasks

Model Inputs: No Embedding



Model Inputs: Word-based Embedding



Using word embeddings alleviates but does not solve the unknown word problem.

Therefore it’s probably a good idea to include features as backup for unknown words (although we can try how well things work without)

Model Inputs: Character-Based Embedding

Useful information about morphology etc. is coded in the vector even for unknown words

embedding similar to “boulevard’' and “avenue"

0.01	0.09	-0.31	0.06	…

word embedding

“rue"

Vector Models

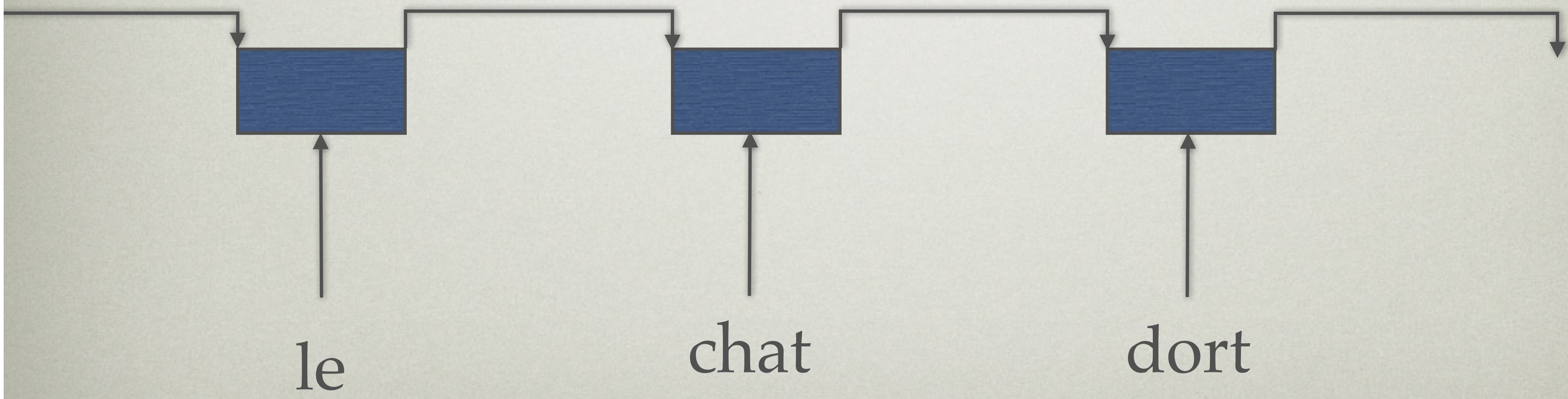
Context-based: word representation depends on context words in the input sentence (not globally in the corpus). “Un avocat mange un avocat”

word2vec	fastText	ELMo	BERT	XLnet
word based	character based			
independent of context		context based		

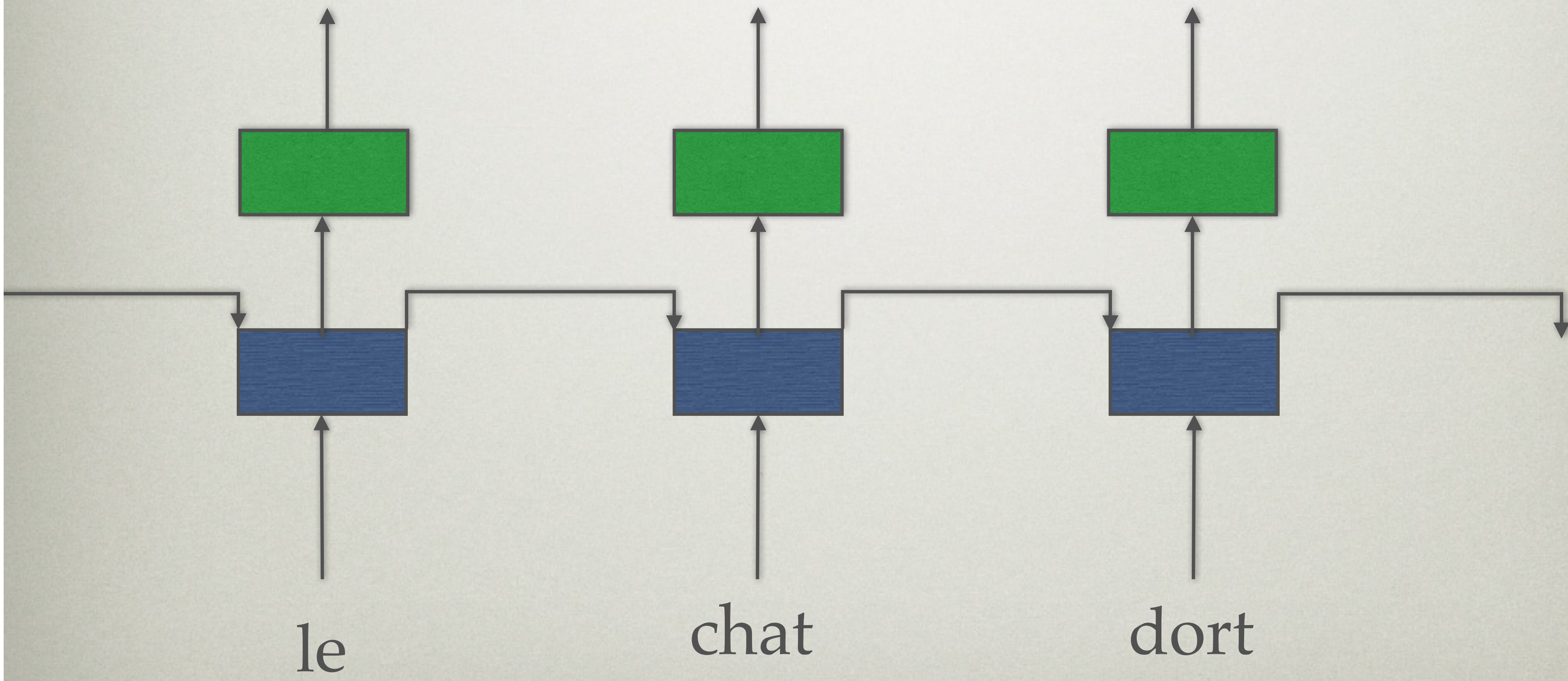
For all character based models, there is a separate neural network which computes the embeddings

Since these models are typically pre-trained on much larger datasets that your corpus, it is usually better to use one of these pre-trained models than to use character-based inputs to your models yourself.

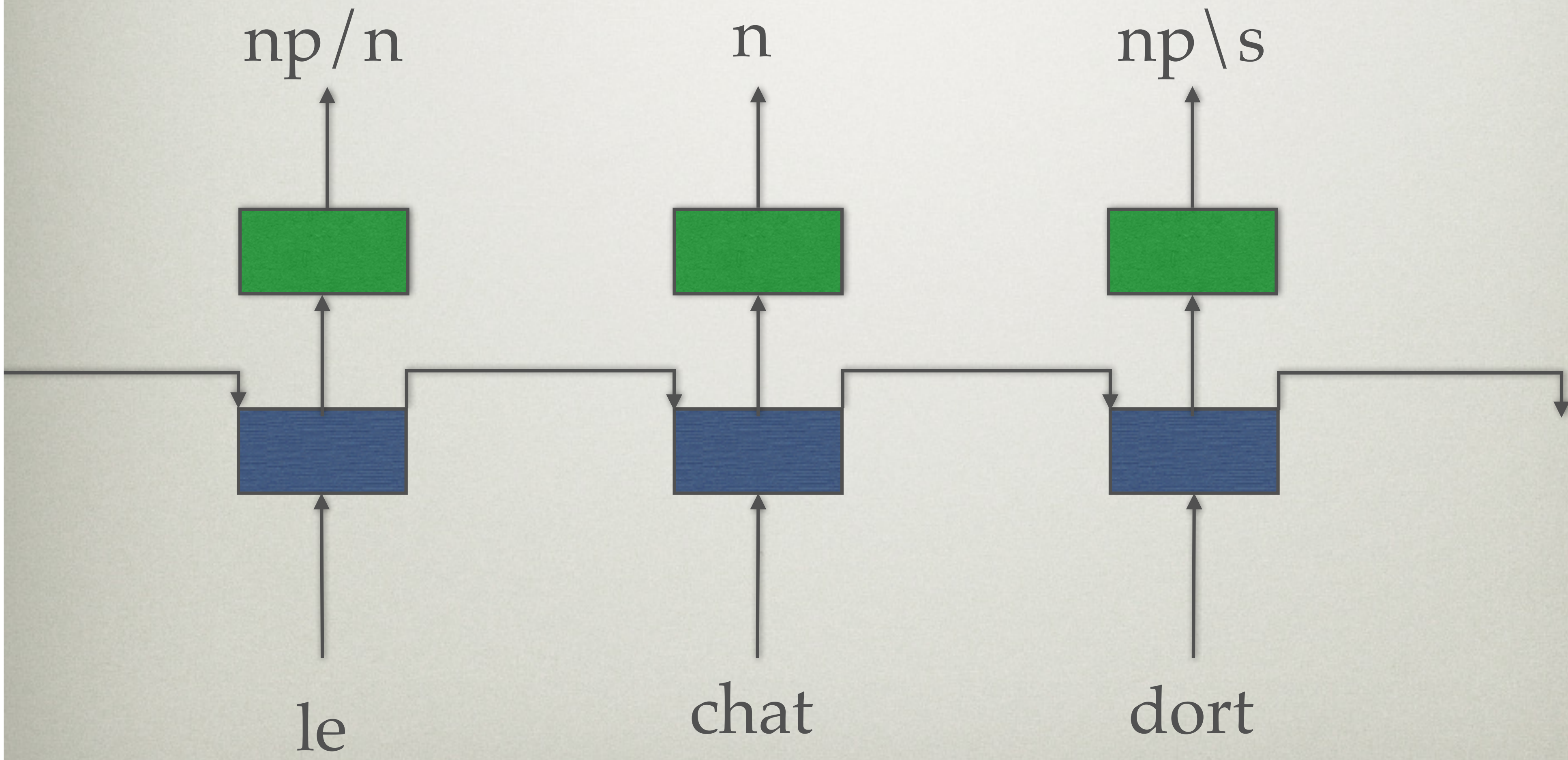
LSTM Tagging/ Supertagging



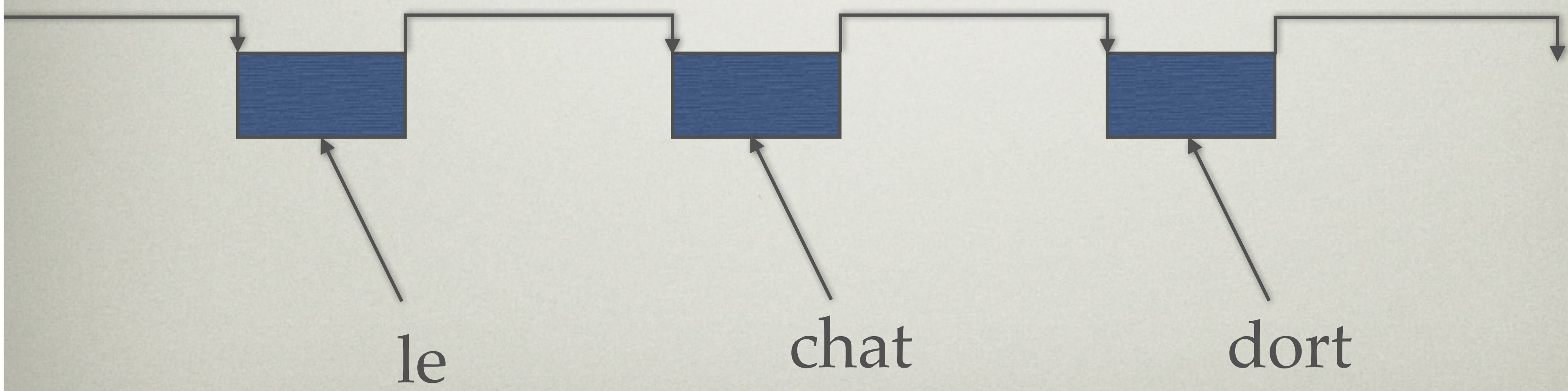
LSTM Tagging/ Supertagging



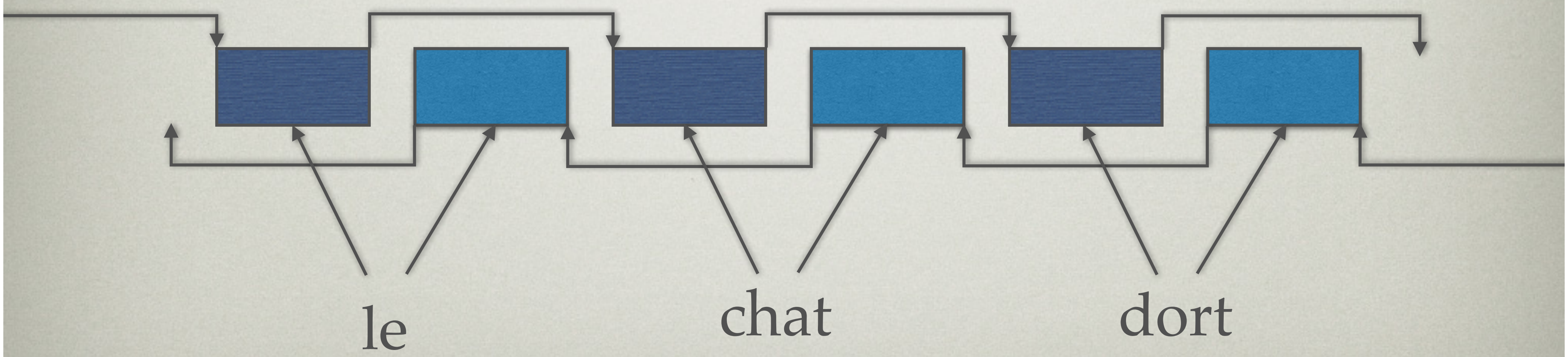
LSTM Tagging/ Supertagging



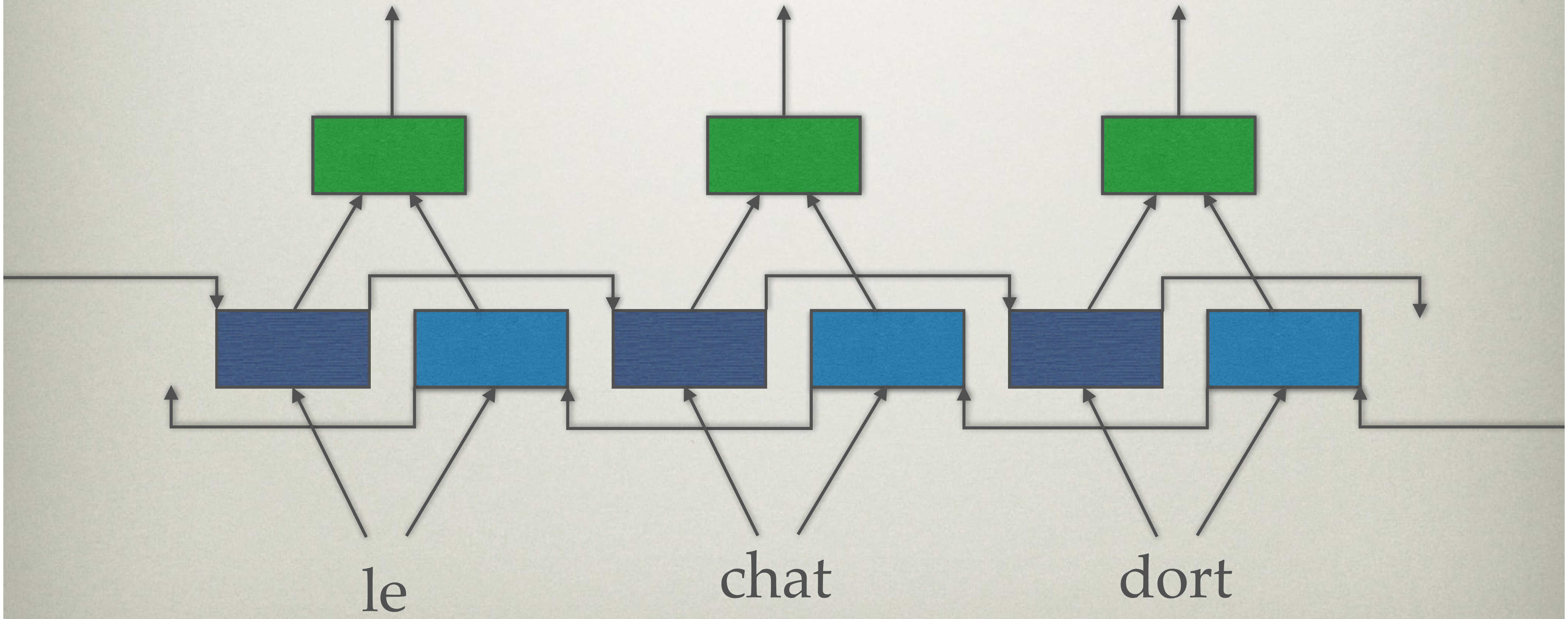
LSTM Tagging/ Supertagging



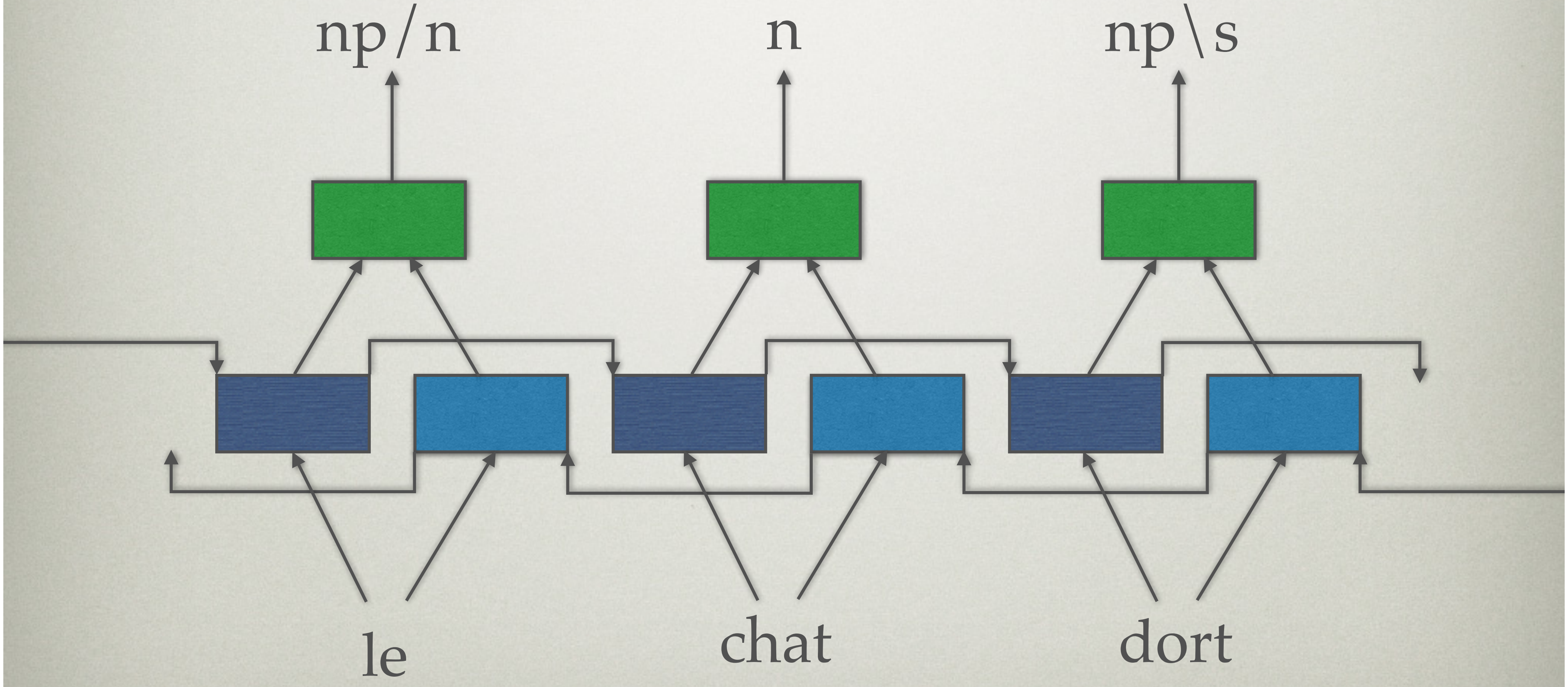
LSTM Tagging/ Supertagging



LSTM Tagging/ Supertagging

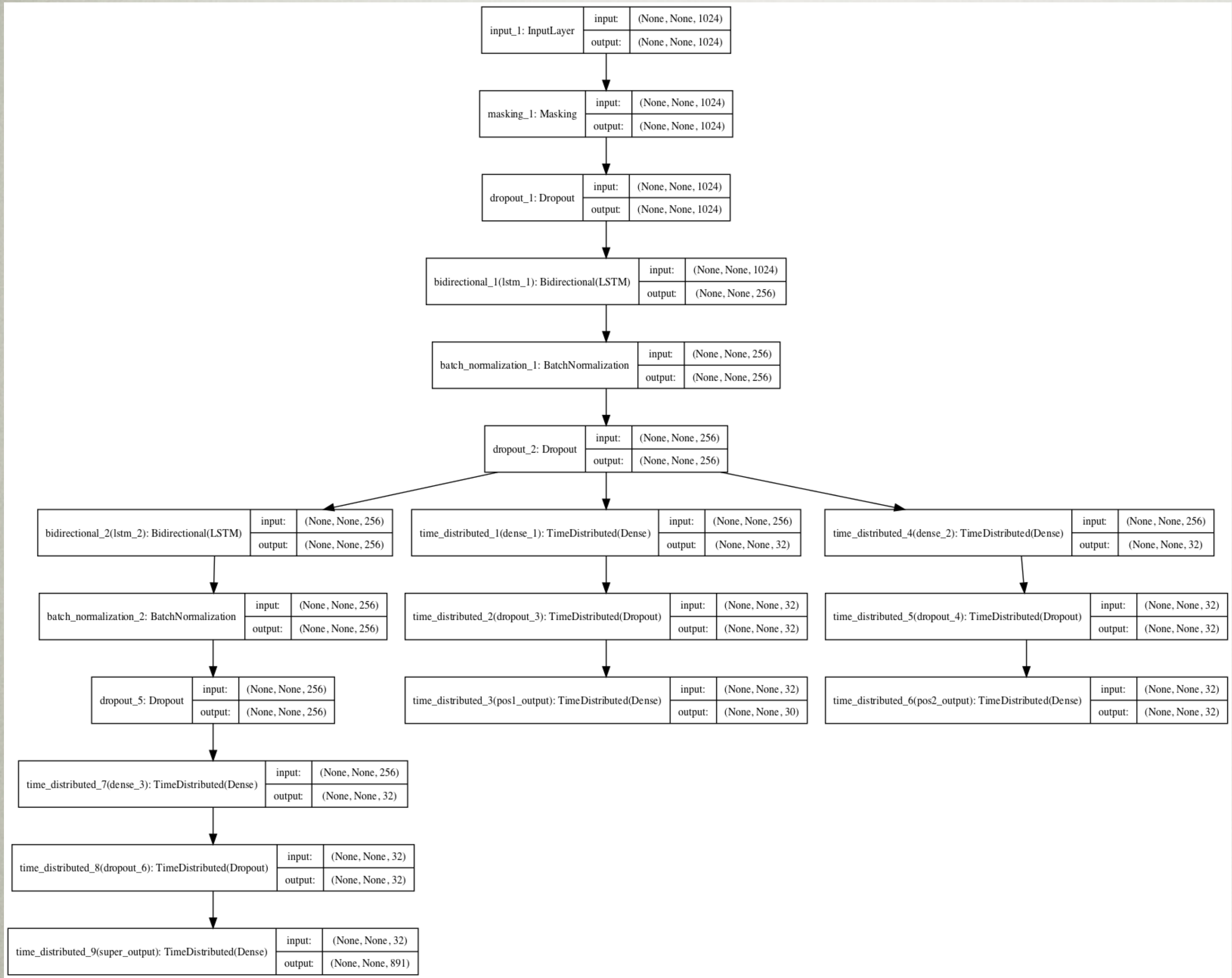


LSTM Tagging/ Supertagging



LSTM Tagging/ Supertagging

```
# input layers are the standard (averaged) ELMo output layer

sentence_embeddings = Input(shape = (None,embLen,), dtype = 'float32')
mask = Masking(mask_value=0.0)(sentence_embeddings)
X = Dropout(0.5)(mask)

# first bi-directional LSTM layer

X = Bidirectional(LSTM(128, recurrent_dropout=0.2, kernel_constraint=max_norm(mxn), return_sequences=True))(X)
X = BatchNormalization()(X)
X = Dropout(dropout_value)(X)

# Pos1 output

Pos1 = TimeDistributed(Dense(32,kernel_constraint=max_norm(mxn)))(X)
Pos1 = TimeDistributed(Dropout(dropout_value))(Pos1)
pos1_output = TimeDistributed(Dense(numPos1Classes, name='pos1_output', activation='softmax',kernel_constraint=max_norm(mxn)))(Pos1)

# Pos2 output

Pos2 = TimeDistributed(Dense(32,kernel_constraint=max_norm(mxn)))(X)
Pos2 = TimeDistributed(Dropout(dropout_value))(Pos2)
pos2_output = TimeDistributed(Dense(numPos2Classes, name='pos2_output', activation='softmax',kernel_constraint=max_norm(mxn)))(Pos2)

# second bi-directional LSTM layer

X = Bidirectional(LSTM(128, recurrent_dropout=dropout_value, kernel_constraint=max_norm(mxn), return_sequences=True))(X)
X = BatchNormalization()(X)
X = Dropout(dropout_value)(X)

# supertag output

X = TimeDistributed(Dense(32, kernel_constraint=max_norm(mxn)))(X)
X = TimeDistributed(Dropout(dropout_value))(X)
super_output = TimeDistributed(Dense(numSuperClasses, name='super_output', activation='softmax', kernel_constraint=max_norm(mxn)))(X)

model = Model(sentence_embeddings, [pos1_output,pos2_output,super_output])
```



The full model has 1,632,057 trainable parameters

Supertagger Performance (MaxEnt)

Corpus	POS	Super	0,1	0,01	F/w
FTB	97,8 %	90,6 %	96,4 %	98,4 %	2,3

Supertagger Performance (MaxEnt)

Corpus	POS	Super	0,1	0,01	F/w
FTB	97,8 %	90,6 %	96,4 %	98,4 %	2,3
Le Monde 2010	97,3 %	89,9 %	95,8 %	97,9 %	2,2
Sequoia/Annodis	97,3 %	88,1 %	94,8 %	97,6 %	2,4
Itipy/Forbes	95,7 %	86,7 %	93,8 %	97,1 %	2,6

How good is this?

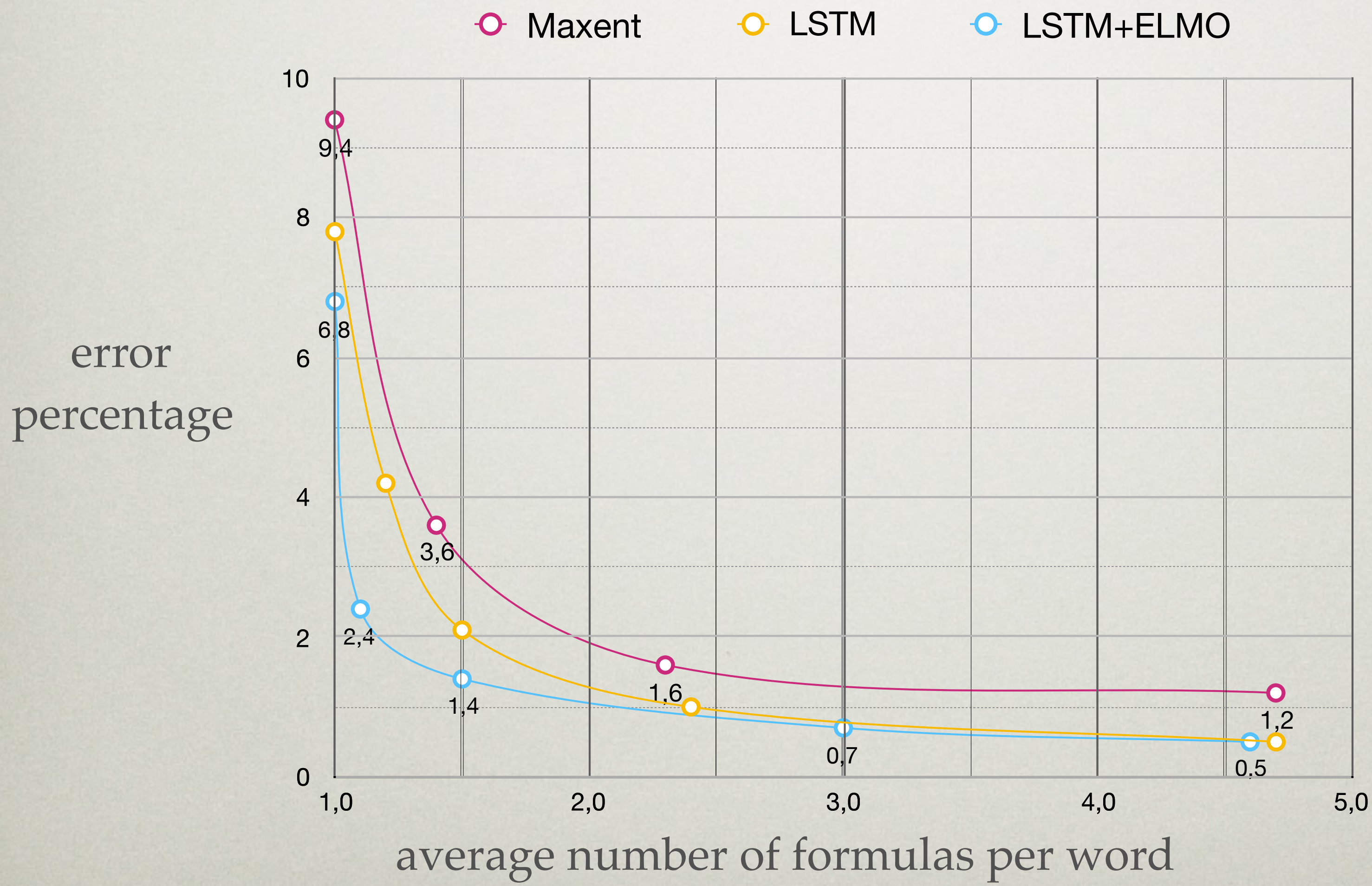
- 90.6% accuracy for the best supertag sounds good, but this is given the correct part-of-speech tag
- When combining POS-tagger with supertagger, accuracy drops to 88.7% (without POS-tagger, we end up at 86.7%, so POS-tagging helps)

Supertagger Performance

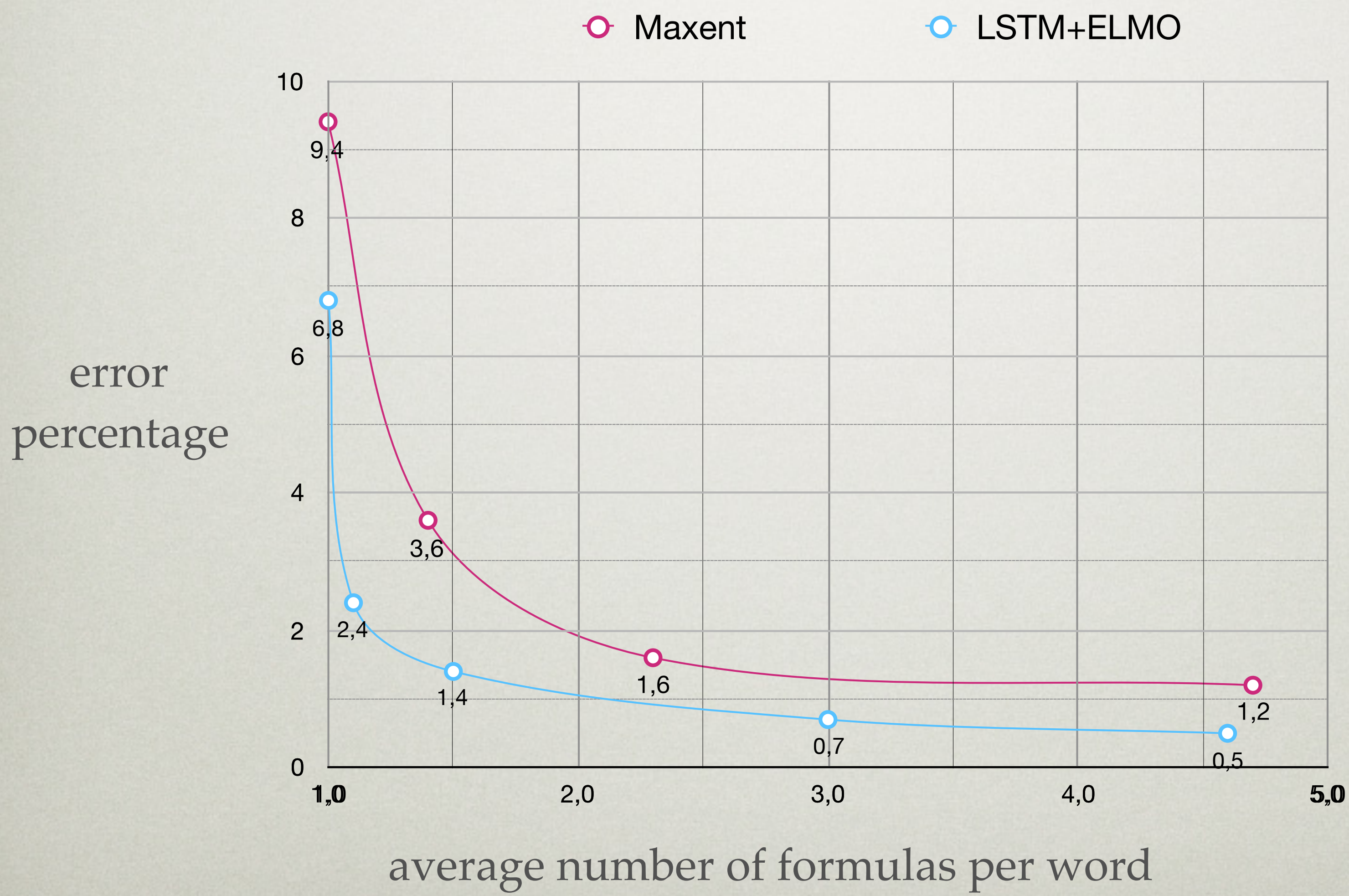
Corpus	POS	Super	0,1	0,01	0,001
MaxEnt	97,8	90,6	96,4 (1,4)	98,4 (2,3)	98,8 (4,7)
LSTM	98,4	92,2	95,8 (1,2)	97,9 (1,5)	99,0 (2,4)
LSTM+ELMo	99,1	93,2	97,6 (1,1)	98,6 (1,5)	99,3 (3,0)

with β=0,0003, we have 4,6 formulas per word (same as ME with β=0,001) but accuracy of 99.5%

LSTM vs MaxEnt



LSTM vs MaxEnt



Taking stock

- Vector representations improve supertagger results.
- When our vector representations are rich enough

1. we no longer need ad hoc features to deal with unknown words
2. our results improve

Moving Vectors Deeper into our Models

- This is a somewhat superficial use of vectors.
- Can vector representations help us choose the “best” proof of a sentence?
- We need two components: a way of *composing* vectors and a way of *evaluating* how good vectors (or combinations of vectors) are.

Proof Search in Natural Deduction

moons	*which*	*Galileo*	*discovered*
n	(n\n)/(s/np)	np	(np\s)/np

n

Proof Search in Natural Deduction

moons		*Galileo*	*discovered*
n		np	(np\s)/np

which

$$\dfrac{n \qquad \dfrac{(n\backslash n)/(s/np) \qquad s/np}{n\backslash n}/E}{n}\backslash E$$

Proof Search in Natural Deduction

Galileo	*discovered*
np	(np\s)/np

$$
\cfrac{\overset{\textit{moons}}{n} \qquad \cfrac{\overset{\textit{which}}{(n\backslash n)/(s/np)} \qquad s/np}{n\backslash n}\,/E}{n}\,\backslash E
$$

Proof Search in Natural Deduction

$$\dfrac{\overset{\textit{moons}}{n} \qquad \dfrac{\overset{\textit{which}}{(n\backslash n)/(s/np)} \qquad \dfrac{\dfrac{\overset{\textit{Galileo}}{np} \qquad \dfrac{\overset{\textit{discovered}}{(np\backslash s)/np} \qquad \overset{[np]_1}{np}}{np\backslash s}\,/E}{s}\,\backslash E}{s/np}\,/I_1}{n\backslash n}\,/E}{n}\,\backslash E$$

Proof Search in Natural Deduction

$$\dfrac{\overset{\textit{moons}}{n} \qquad \dfrac{\overset{\textit{which}}{(n\backslash n)/(s/np)} \qquad \dfrac{\dfrac{\overset{\textit{Galileo}}{np} \qquad \dfrac{\overset{\textit{discovered}}{(np\backslash s)/np} \quad [np]_1}{np\backslash s}\,/E}{s}\,\backslash E}{s/np}\,/I_1}{n\backslash n}\,/E}{n}\,\backslash E$$

Natural Deduction With Vectors

moons	*which*	*Galileo*	*discovered*
n	(n\n)/(s/np)	np	(np\s)/np

When the goal formula is atomic, we need to *select* the focused formula and *split* the antecedent

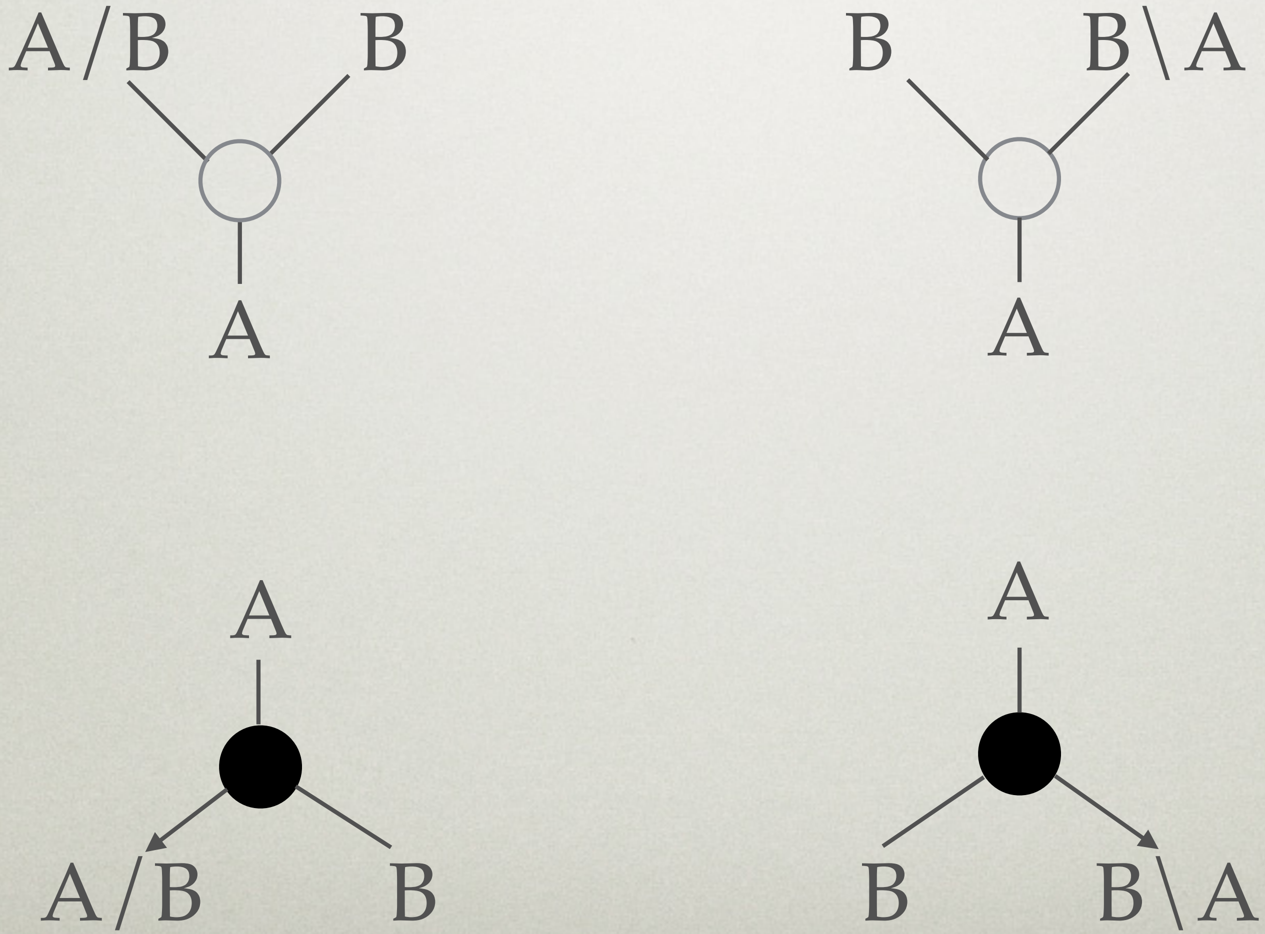
Representing words as vectors and formulas as vectors a neural network can learn these tasks (Kogkalidis e.a. 2019)

This sidesteps the need for vector composition!

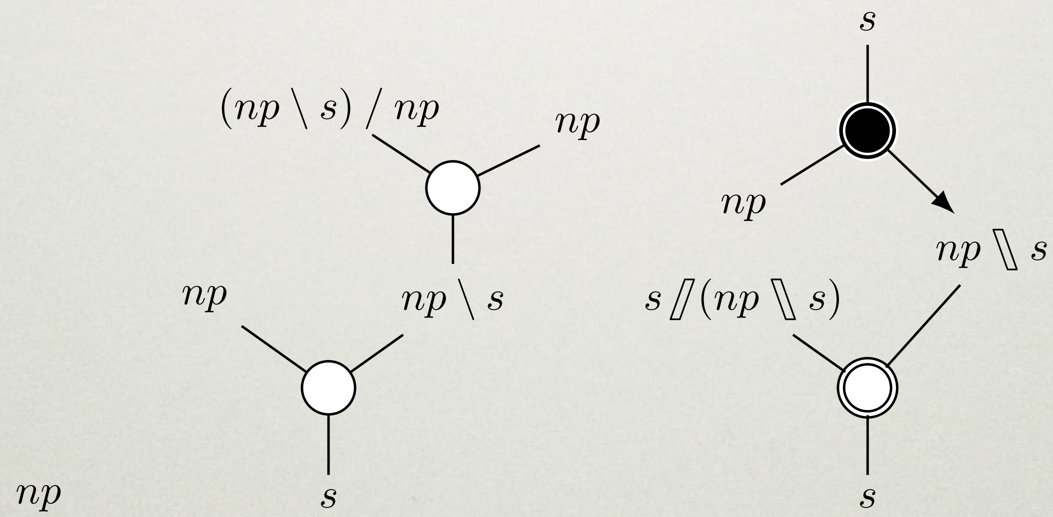
Parsing With Proof Nets

- Proof nets are graph-based proof systems for linear logic and type-logical grammars
- They represent the combinatorics of the search space of proofs in a way which is neutral with respect to any proof search strategy

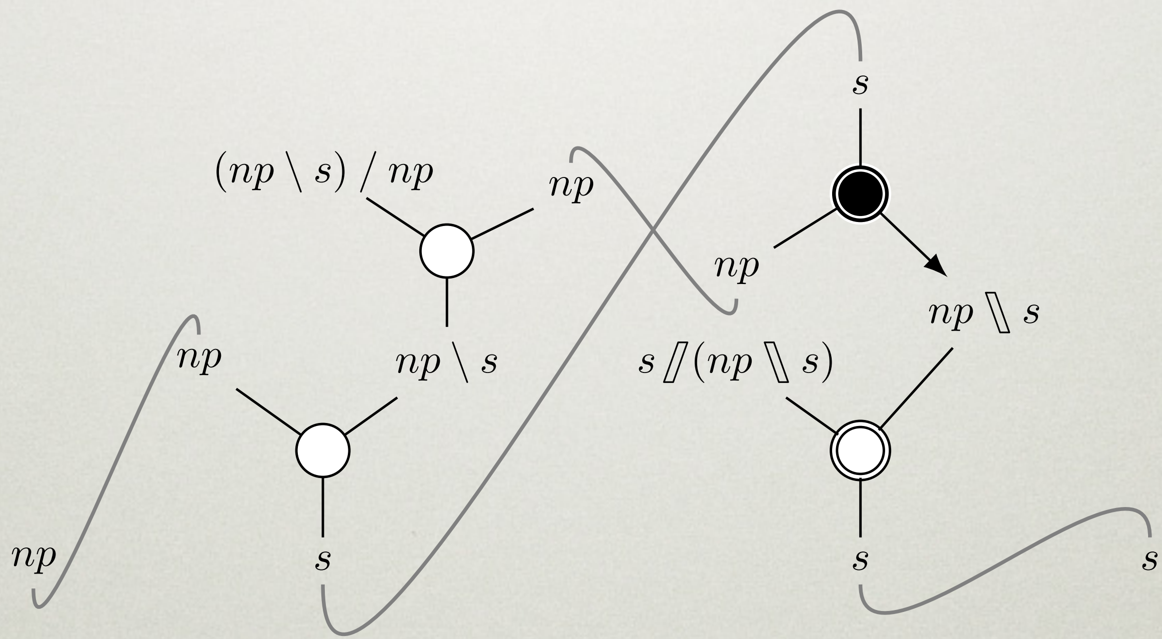
Parsing With Proof Nets



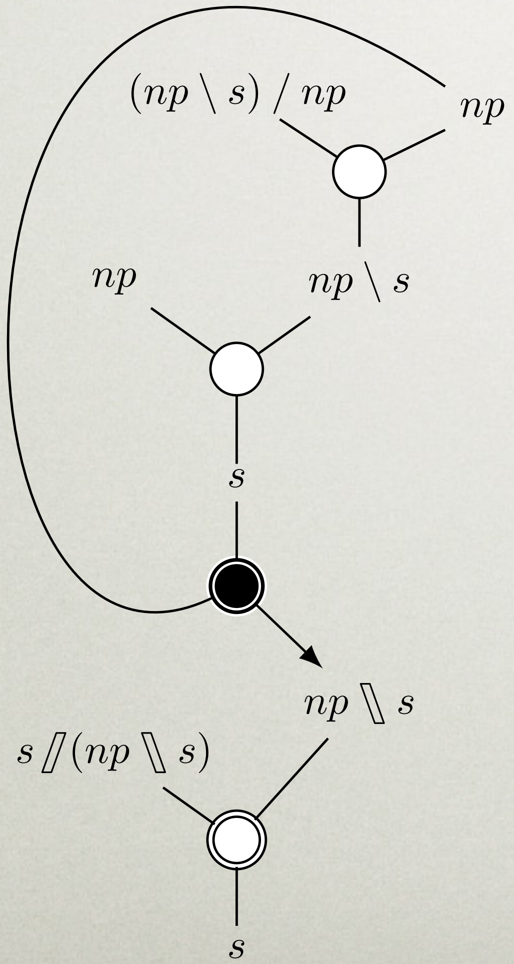
Example: "John saw everyone"



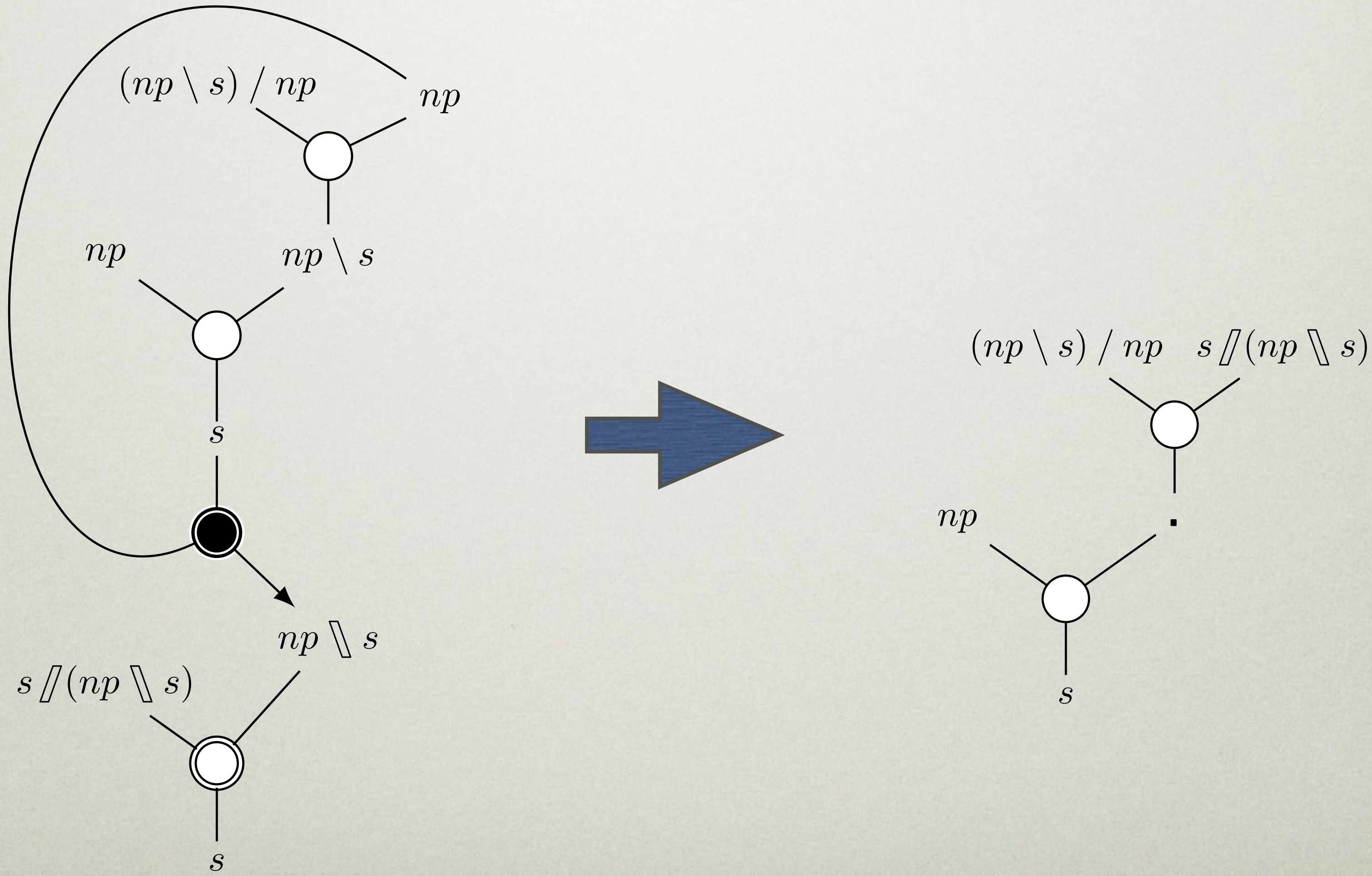
Example: "John saw everyone"



Example: "John saw everyone"



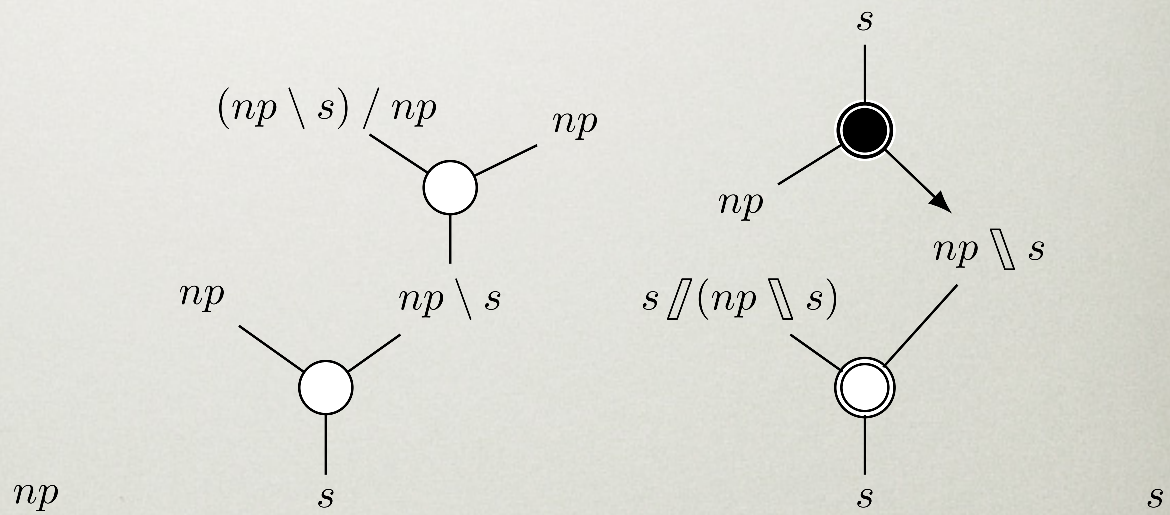
Example: "John saw everyone"



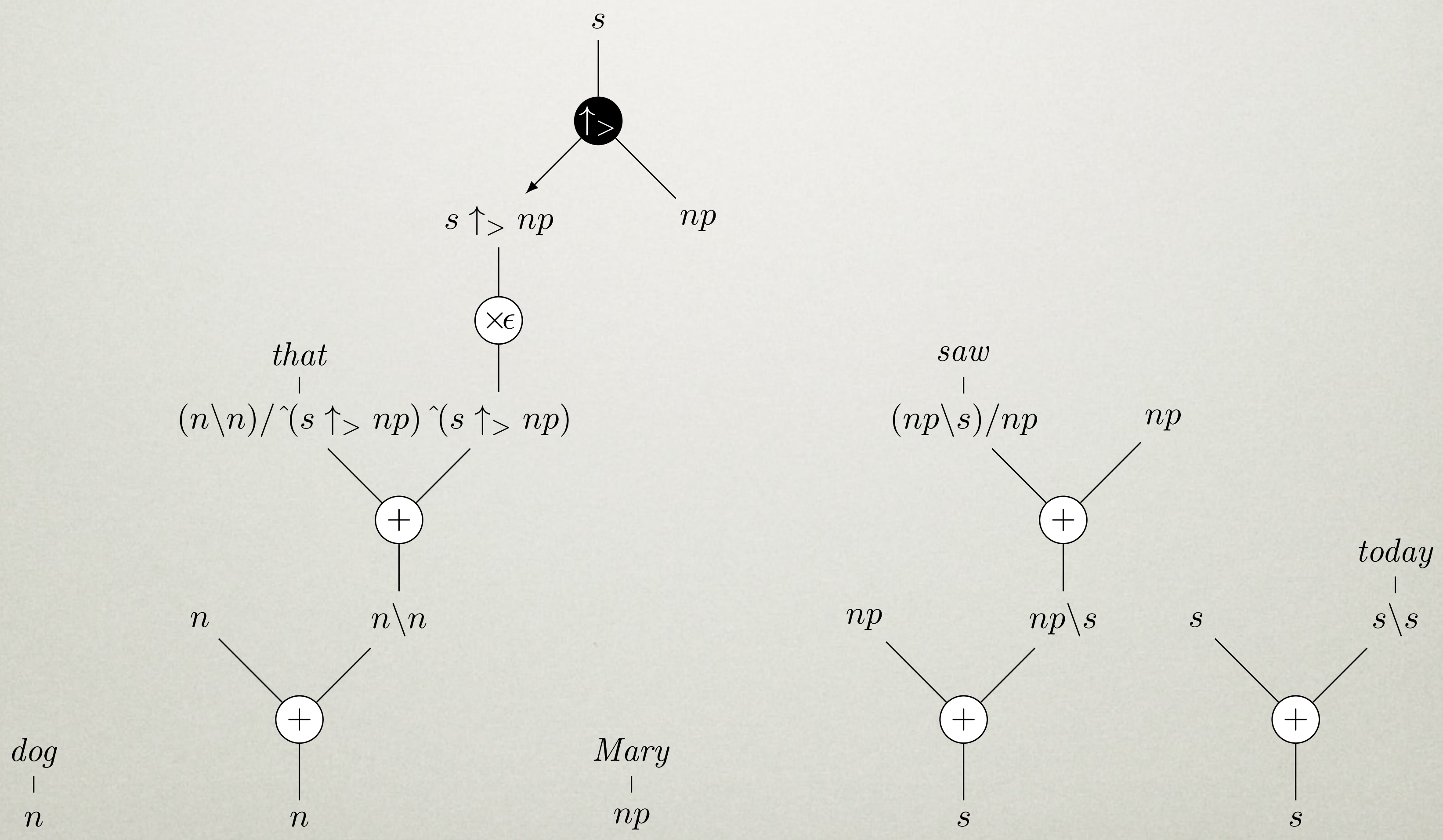
Example: "John saw everyone"

	np^+	np^+
np^-		
np^-		

	s^+	s^+
s^-		
s^-		



"dog that Mary Saw Today"



Propagating Vectors Through Proof Nets

- Specify propagation rules in a generic form with a composition operation and an identity element
- Allow certain lexical entries to override default
- Let neural network learn propagation which helps parsing best

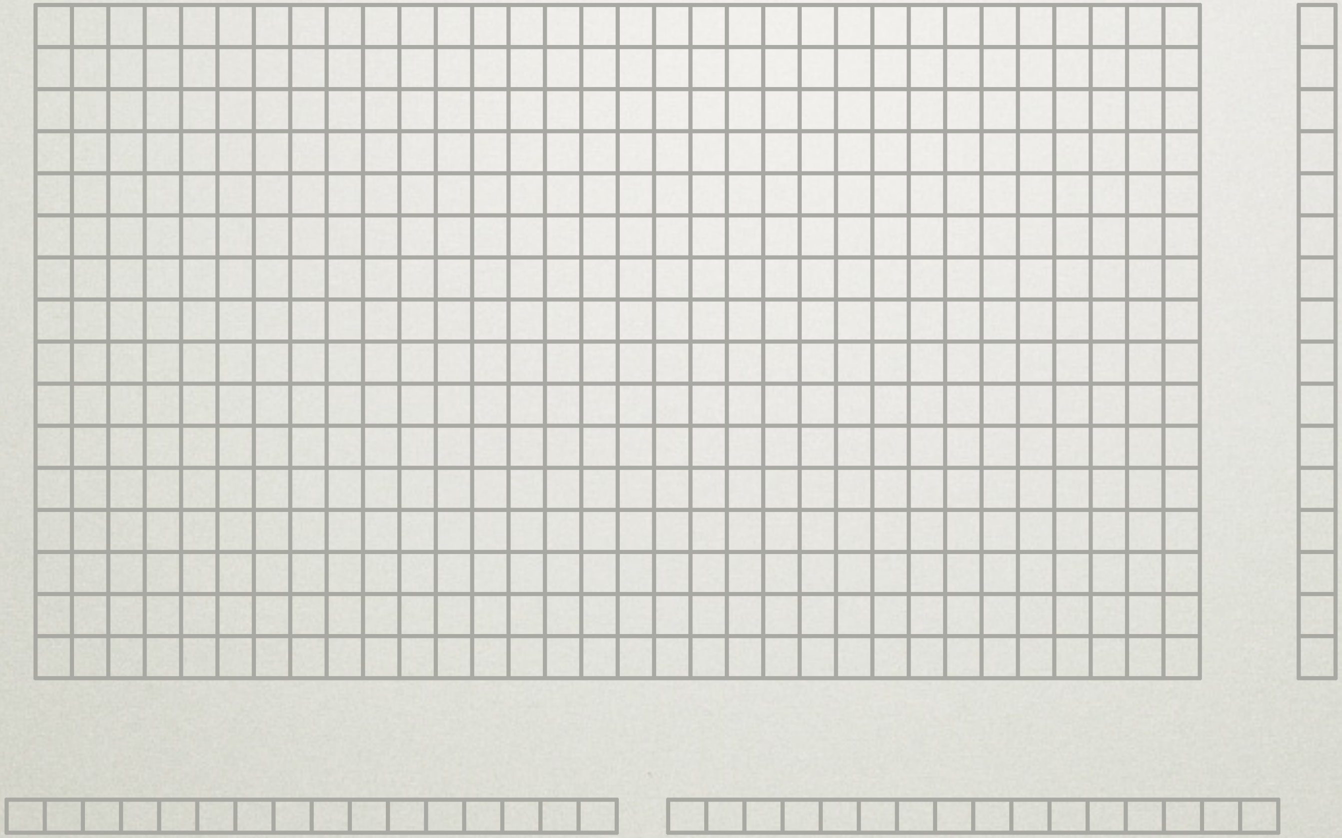
Propagating Vectors Through Proof Nets

- Specify propagation rules in a generic form with a composition operation and an identity element
- Allow certain lexical entries to override default
- Let neural network learn propagation which helps parsing best

Baseline

- Baseline: composition is vector addition (maybe averaged at the end)
- Identity element is zero vector

Composition = Neural Net

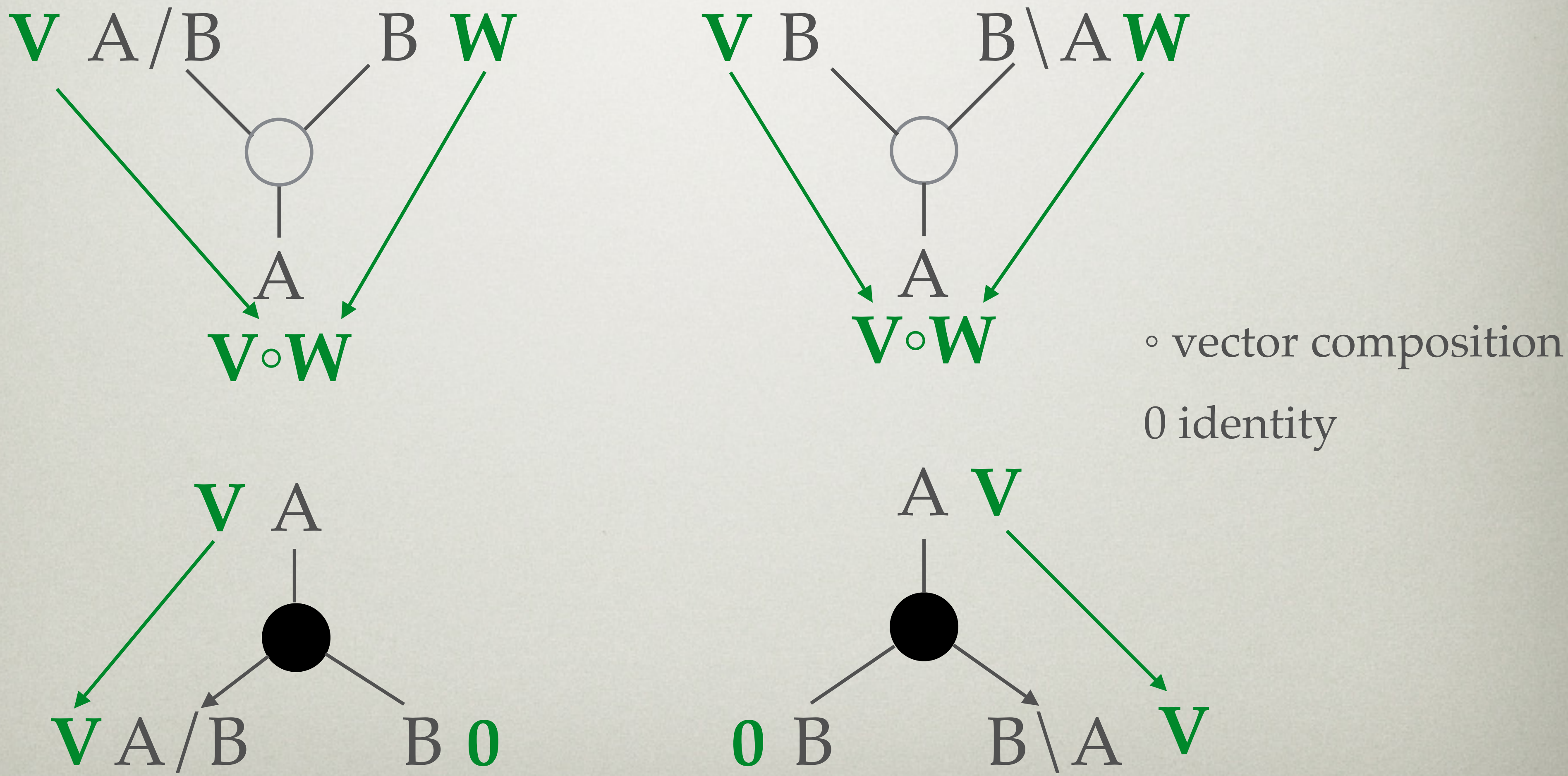


Output vector

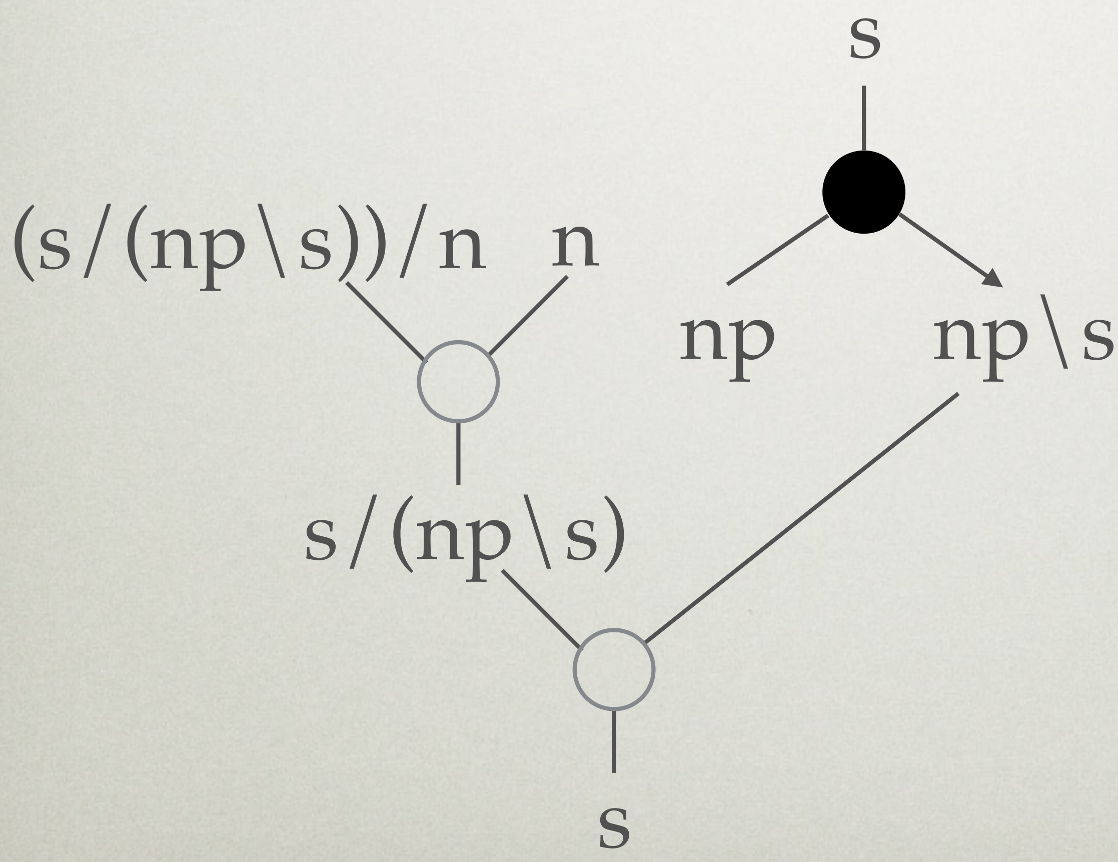
Input vector 1 Input vector 2

Socher, Lin, Ng & Manning (2011)

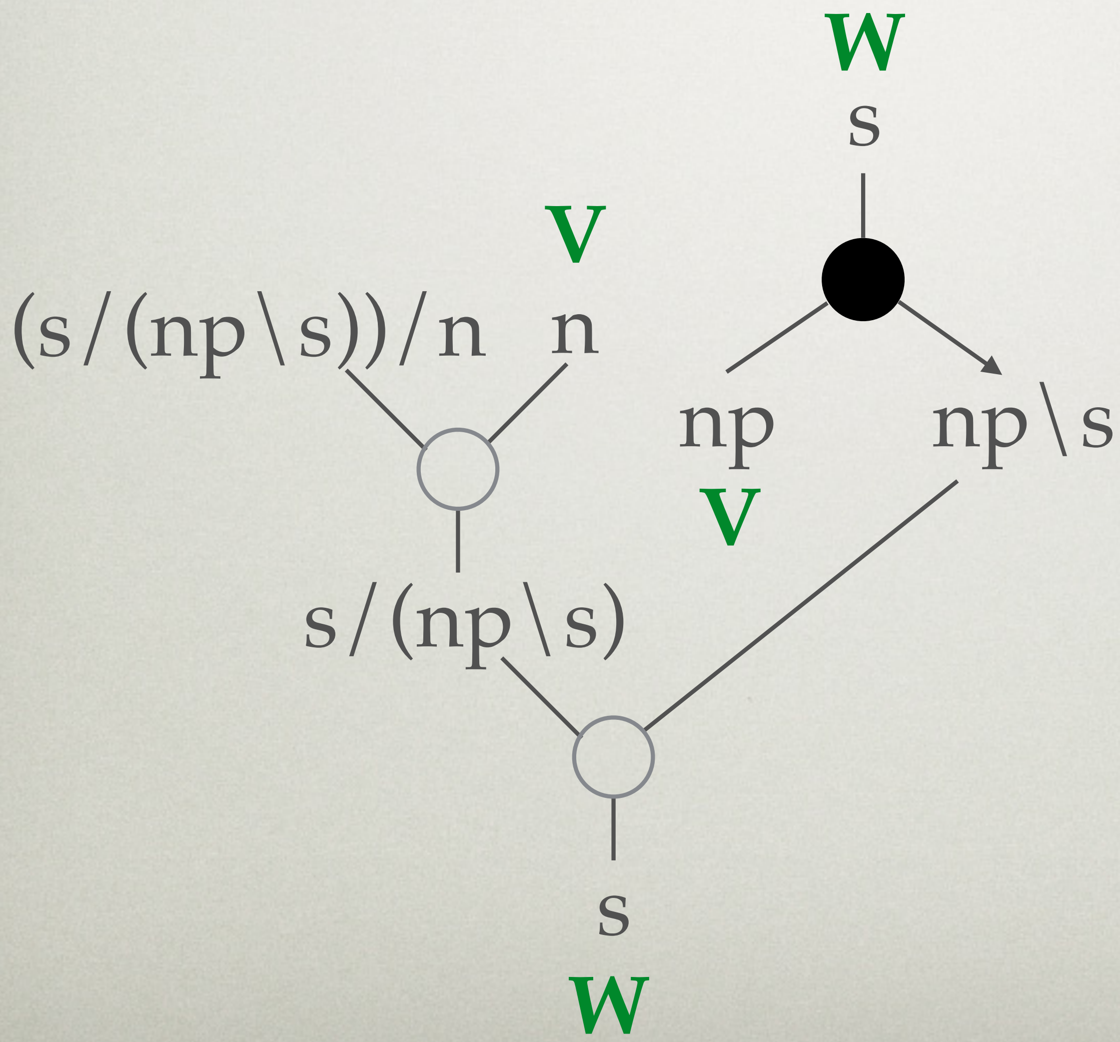
Parsing With Proof Nets



Parsing With Proof Nets



Parsing With Proof Nets



Conclusions

- We have seen several superficial ways of incorporating vector representations and deep learning into type-logical grammars
- For the moment, only the supertagger vectors have been evaluated; they result in a cleaner, less ad hoc model and improved performance

The Future

- We want to evaluate the use of vectors into selecting the best proof and thereby the best lambda term meaning
- Is it useful to incorporate vector semantics for a fully semantic task (eg. entailment) with type-logical grammars?