Adaptive Supertagging [Clark & Curran, 2007]

Start with an initial prob. cutoff β

$$\begin{array}{cccc} \text{He} & \text{reads} & \text{the} & \text{book} \\ \hline \mathsf{NP} & (\mathsf{S}[\mathsf{pss}]\backslash \mathsf{NP})/\mathsf{NP} & \mathsf{NP}/\mathsf{N} & \mathsf{N} \end{array}$$

Adaptive Supertagging [Clark & Curran, 2007]

Prune a category, if its probability is below β times the prob. of the best category

$$\begin{array}{cccc} \text{He} & \text{reads} & \text{the} & \text{book} \\ \hline \mathit{NP} & (S[\mathit{pss}]\backslash \mathit{NP})/\mathit{NP} & \mathit{NP}/N & N \end{array}$$

Adaptive Supertagging [Clark & Curran, 2007]

Decrease β if no spanning analysis

He	reads	the	book
NP	$(S[pss]\backslash NP)/NP$	NP/N	N
N	$(S\backslash NP)/NP$	NP/NP	$(S\backslash NP)/NP$
N/N	$S\backslash NP$		

Adaptive Supertagging [Clark & Curran, 2007]

Decrease β if no spanning analysis

He	reads	the	book
NP	$(S[pss]\backslash NP)/NP$	NP/N	N
N	$(S\backslash NP)/NP$	NP/NP	$(S\backslash NP)/NP$
N/N	$S\backslash NP$		
NP/NP	$(S[pt]\backslash NP)/NP$		
	$(S[dcl]\backslash NP)/NP$		

Recurrent neural networks (RNN)

Recurrent neural networks

- Use the same computational function and parameters across different time steps of the sequence
- Each time step: takes the input entry and the previous hidden state to compute the output entry
- Loss: typically computed every time step

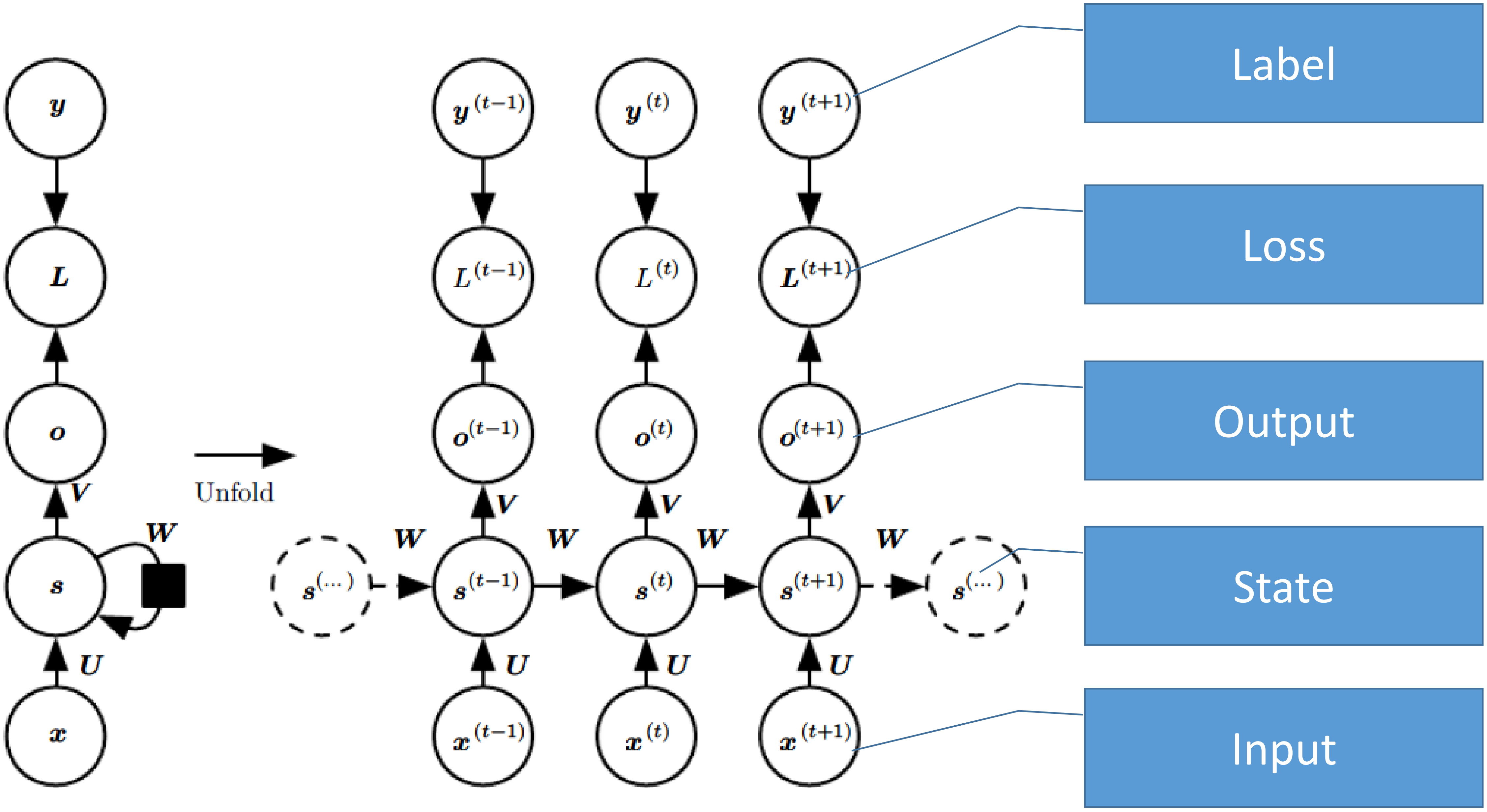
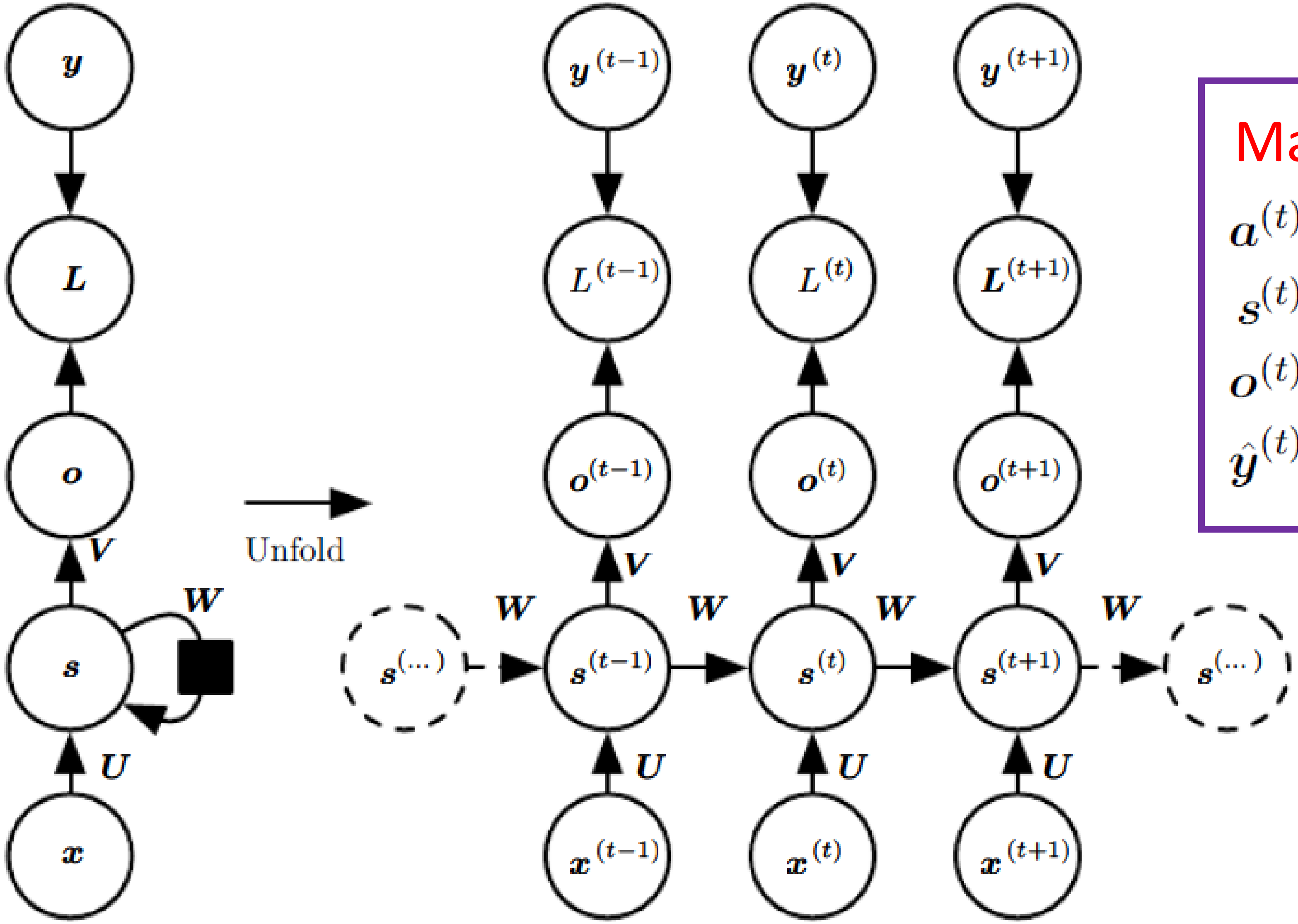


Figure from *Deep Learning*, by Goodfellow, Bengio and Courville



Math formula:

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{s}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{s}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{s}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}
$$

Figure from *Deep Learning*, Goodfellow, Bengio and Courville

Advantage

- Hidden state: a lossy summary of the past
- Shared functions and parameters: greatly reduce the capacity and good for generalization in learning
- Explicitly use the prior knowledge that the sequential data can be processed by in the same way at different time step (e.g., NLP)

Advantage

- Hidden state: a lossy summary of the past
- Shared functions and parameters: greatly reduce the capacity and good for generalization in learning
- Explicitly use the prior knowledge that the sequential data can be processed by in the same way at different time step (e.g., NLP)

- Yet still powerful (actually universal): any function computable by a Turing machine can be computed by such a recurrent network of a finite size (see, e.g., Siegelmann and Sontag (1995))

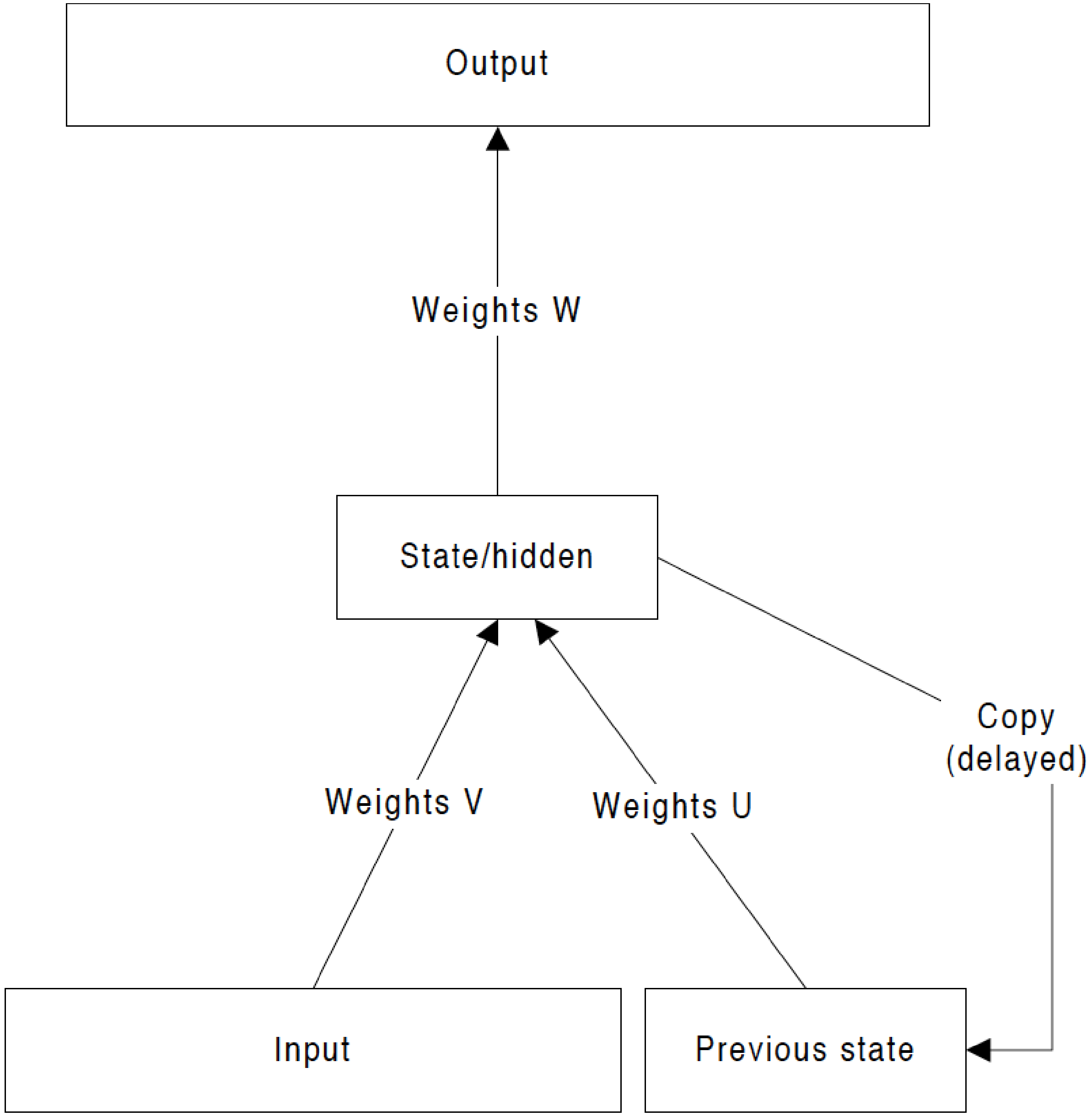
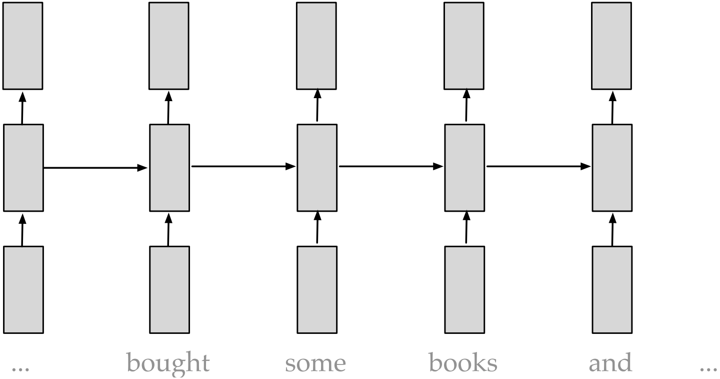


Figure 4: A simple recurrent network.

Supertagging with a RNN

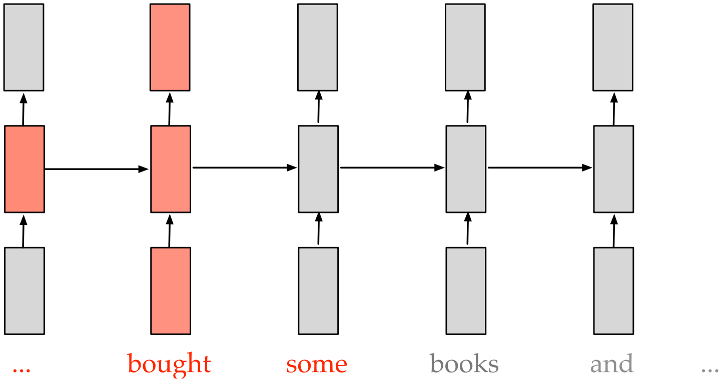
- Using only dense features
  - word embedding
  - suffix embedding
  - capitalization
- The input layer is a concatenation of all embeddings of all words in a context window

Supertagging with a RNN

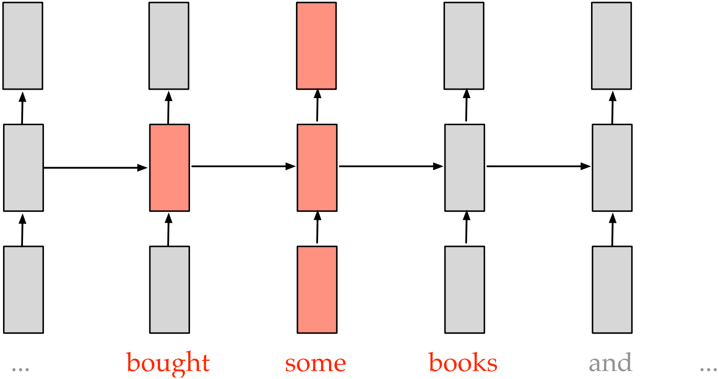


... bought some books and ...

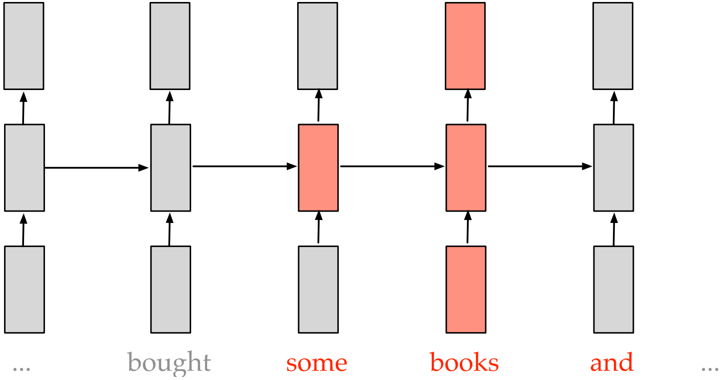
Supertagging with a RNN



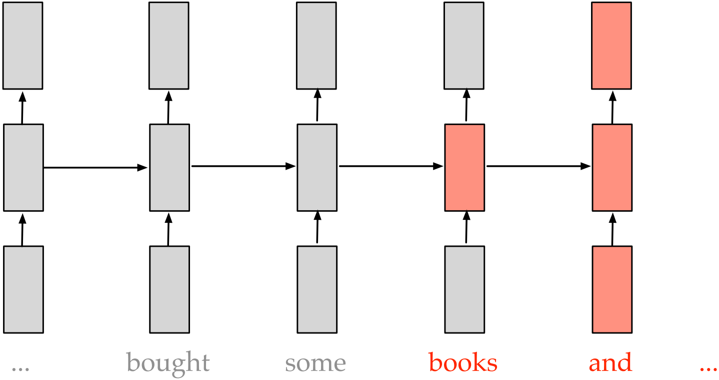
Supertagging with a RNN



Supertagging with a RNN



Supertagging with a RNN



1-best Supertagging Results: dev

Model	Accuracy	Time
C&C (gold POS)	92.60	-
C&C (auto POS)	91.50	0.57
NN	91.10	21.00
RNN	92.63	-
RNN+dropout	93.07	2.02

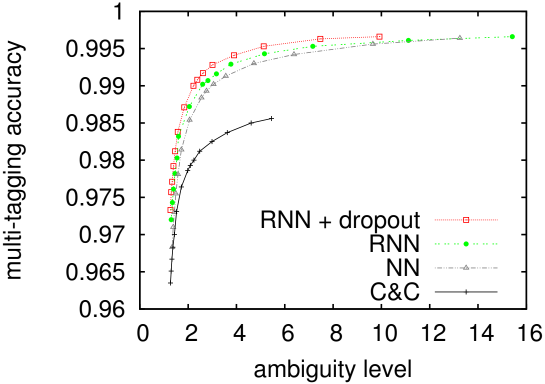
Table 1 : 1-best tagging accuracy and speed comparison on CCGBank Section 00 with a single CPU core (1,913 sentences), tagging time in secs.

1-best Supertagging Results: test

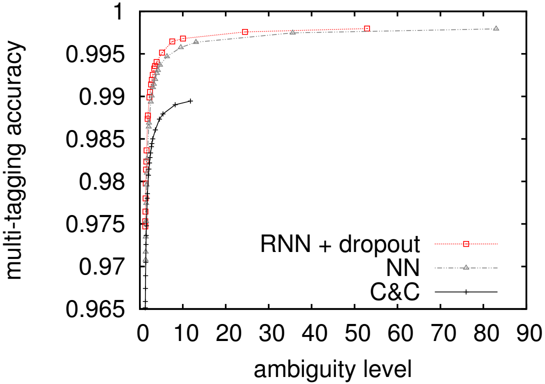
Model	Section 23	Wiki	Bio
C&C (gold POS)	93.32	88.80	91.85
C&C (auto POS)	92.02	88.80	89.08
NN	91.57	89.00	88.16
RNN	93.00	90.00	88.27

Table 2 : 1-best tagging accuracy comparison on CCGBank Section 23 (2,407 sentences), Wikipedia (200 sentences) and Bio-GENIA (1,000 sentences).

Multi-tagging Results: dev



Multi-tagging Results: test



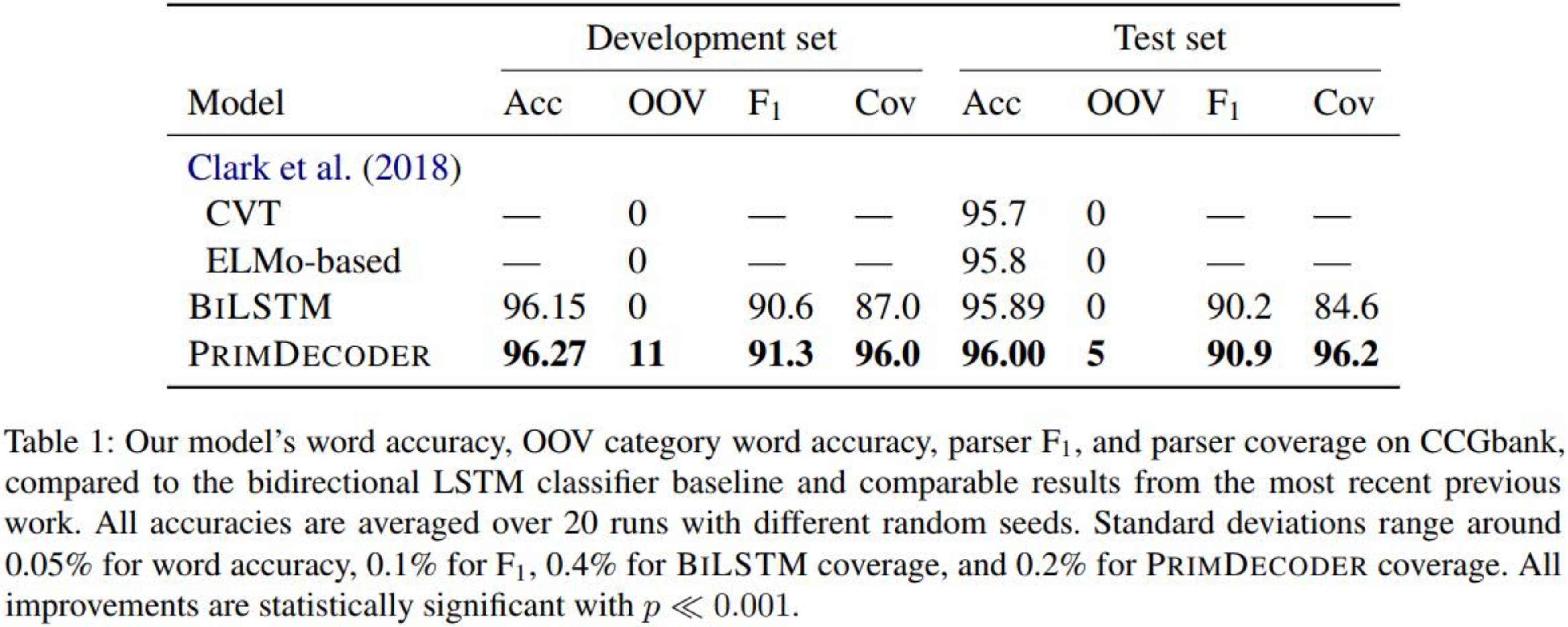
Final Parsing Results

	CCGBank Section 23				Wikipedia			
	LP	LR	LF	cov.	LP	LR	LF	
C&C	86.24	84.85	85.54	99.42	81.58	80.08	80.83	99.50
(NN)	86.71	85.56	86.13	99.92	82.65	81.36	82.00	**100**
(RNN)	**87.68**	**86.47**	**87.07**	**99.96**	**83.22**	**81.78**	**82.49**	**100**
C&C	86.24	84.17	85.19	100	81.58	79.48	80.52	100
(NN)	86.71	85.40	86.05	100	-	-	-	-
(RNN)	**87.68**	**86.41**	**87.04**	100	-	-	-	-

Table 3 : Parsing test results (auto POS). We evaluate on all sentences (100% coverage) as well as on only those sentences that returned spanning analyses (% cov.). RNN and NN both have 100% coverage on the Wikipedia data.

Training & Experiments

- Mini-batched BPTT [Rumelhart et al., 1988; Mikolov, 2012]
- A context window-size of 7, a BPTT step size of 9, avg ambig 1.4
- 50-dim scaled Turian embeddings [Turian et al., 2010]
- Other two look-up tables randomly initialized
- Embedding fine-tuning during training
- Dropout regularization
- Parsing experiments: use the same supertagger prob. cutoff values as C&C



	Development set				Test set			
Model	Acc	OOV	F_1	Cov	Acc	OOV	F_1	Cov
Clark et al. (2018)								
CVT	—	0	—	—	95.7	0	—	—
ELMo-based	—	0	—	—	95.8	0	—	—
BiLSTM	96.15	0	90.6	87.0	95.89	0	90.2	84.6
PrimDecoder	**96.27**	**11**	**91.3**	**96.0**	**96.00**	**5**	**90.9**	**96.2**

Table 1: Our model's word accuracy, OOV category word accuracy, parser F_1, and parser coverage on CCGbank, compared to the bidirectional LSTM classifier baseline and comparable results from the most recent previous work. All accuracies are averaged over 20 runs with different random seeds. Standard deviations range around 0.05% for word accuracy, 0.1% for F_1, 0.4% for BiLSTM coverage, and 0.2% for PrimDecoder coverage. All improvements are statistically significant with $p \ll 0.001$.