Sequence Tagger

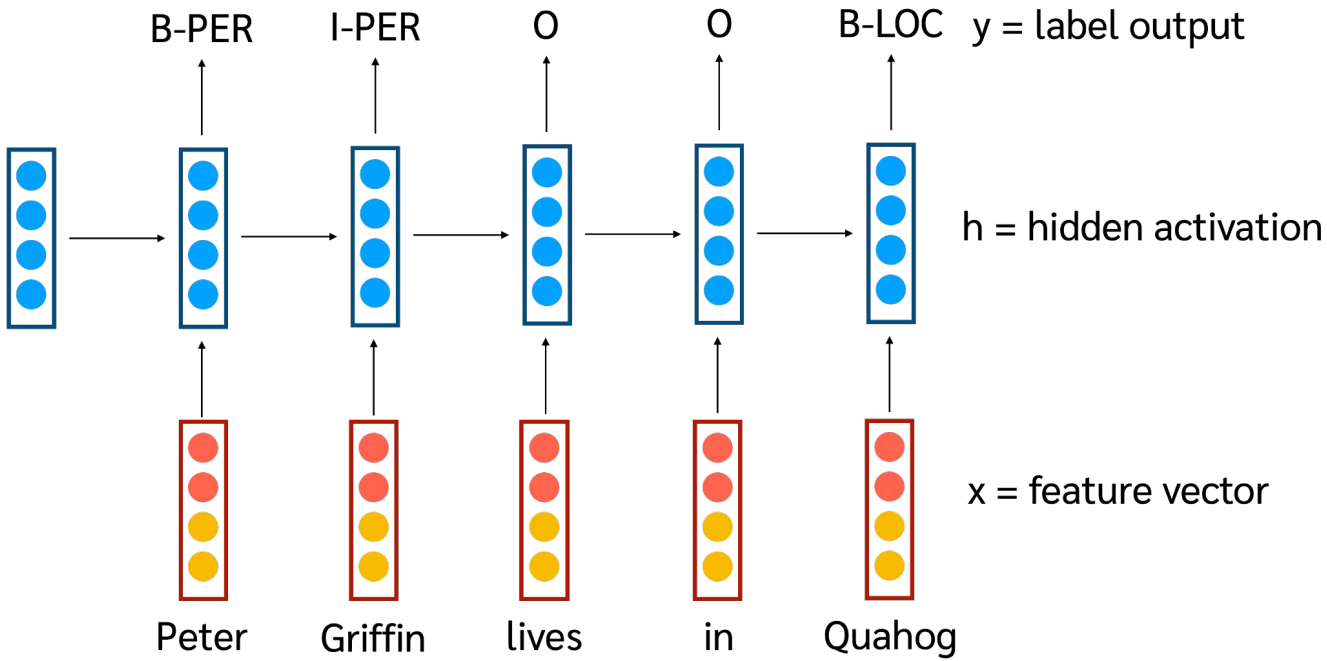
NLP II 2025

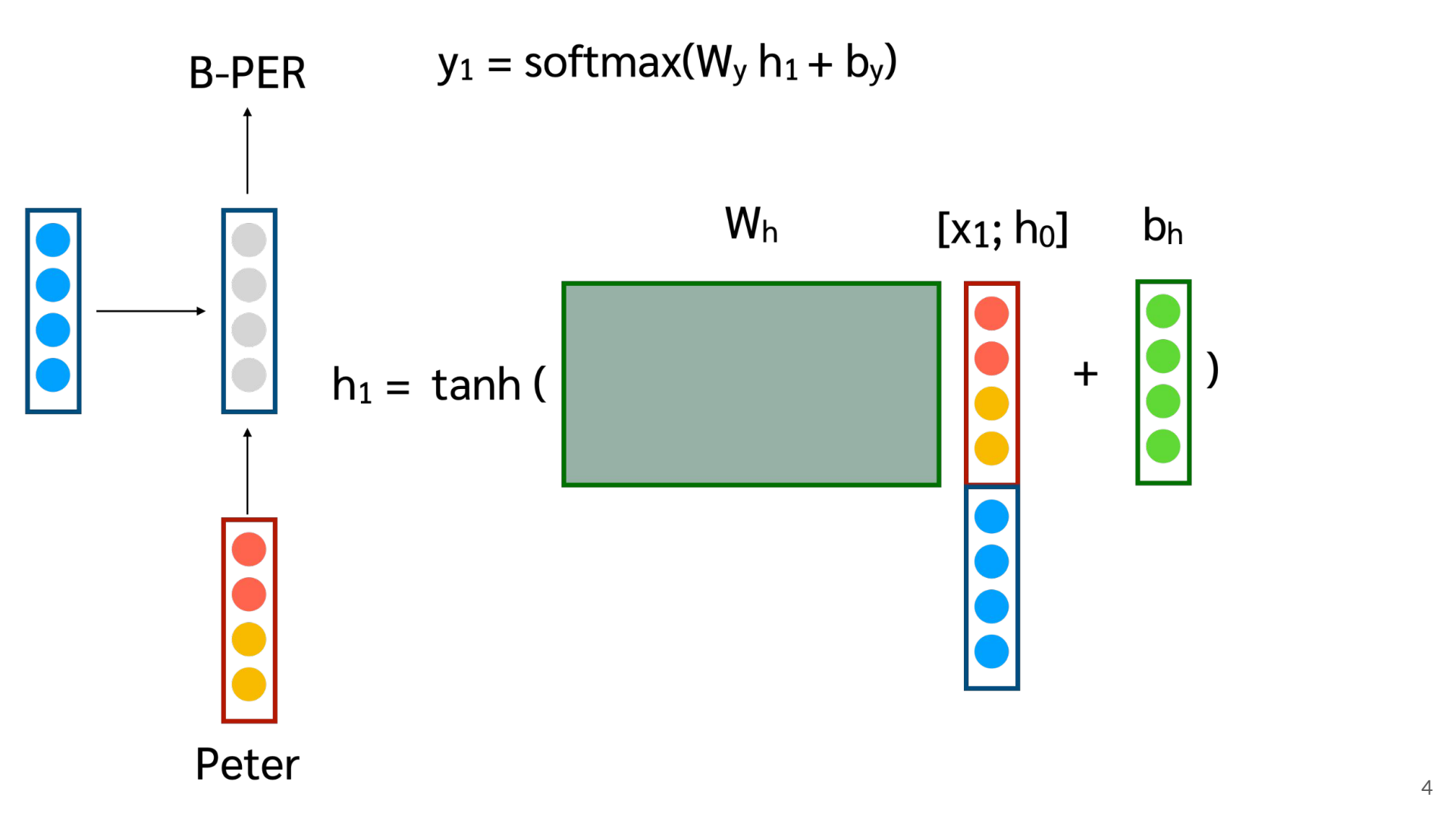
Jakapun Tachaiya (Ph.D.)

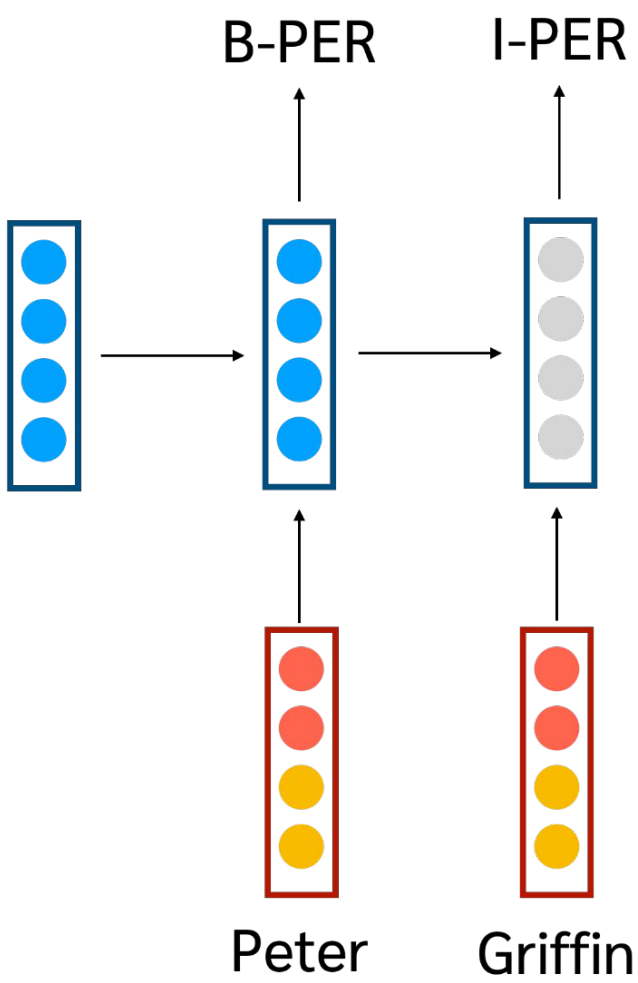
Recurrent Neural Network

- The model is exactly the same as the one we use for language model (predicting the next word for each token), but now we use it as a sequence tagger (predicting the label for each token)

Recurrent Neural Network

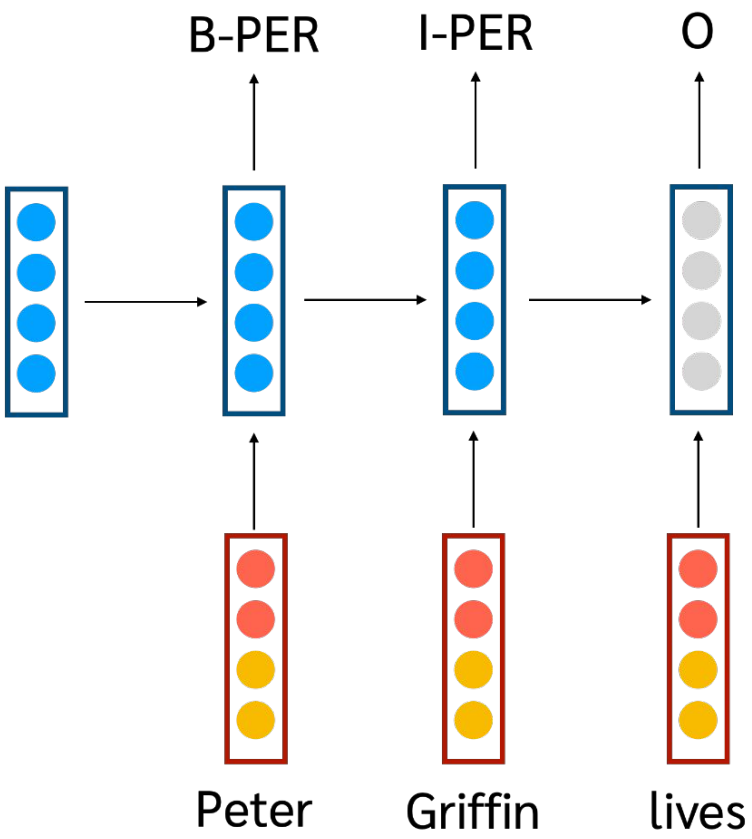



B-PER
$y_1 = \text{softmax}(W_y h_1 + b_y)$
W_h
$[x_1; h_0]$
b_h
$h_1 = \tanh($
+
)
Peter



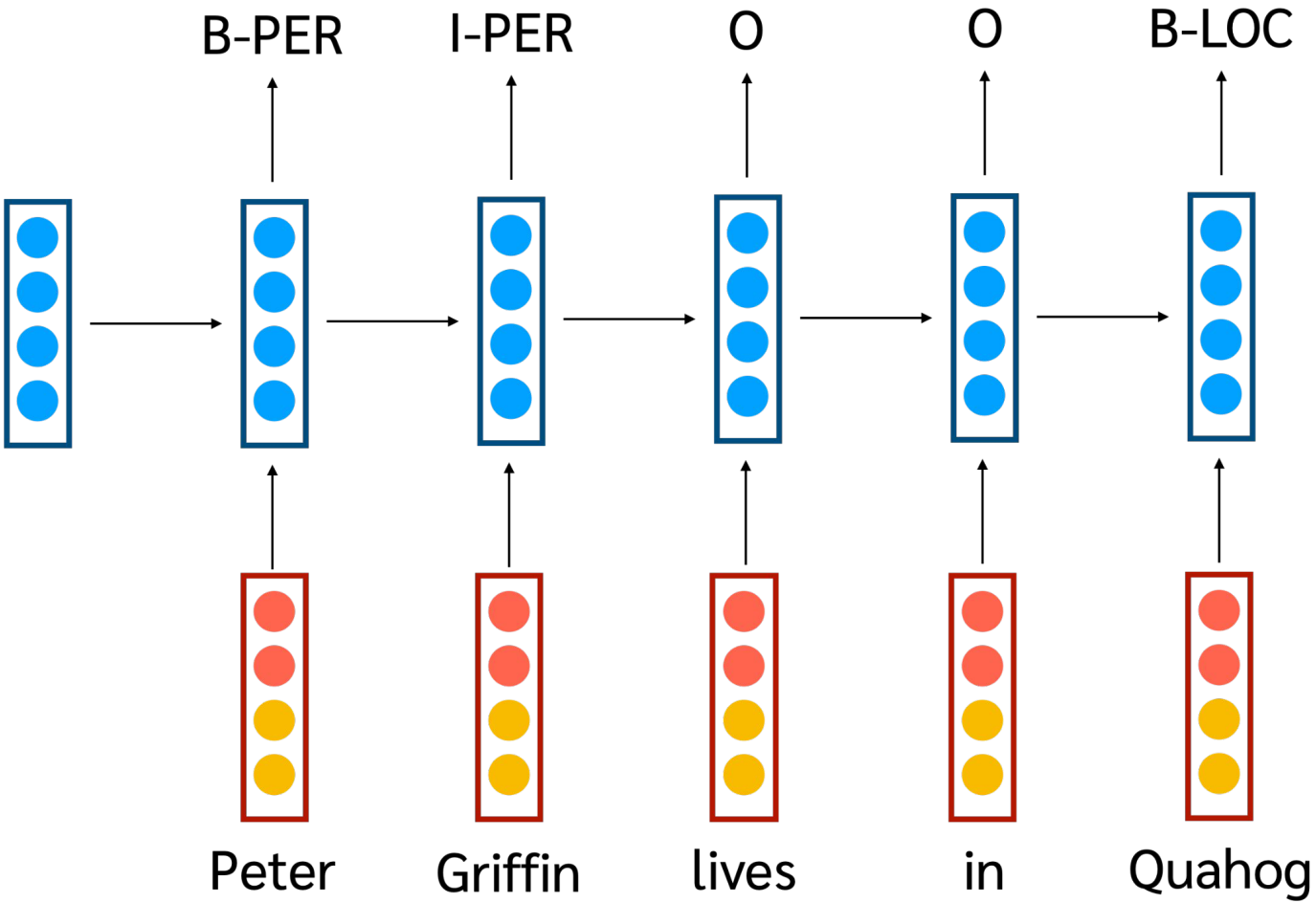
$$y_2 = \text{softmax}(W_y h_2 + b_y)$$

$$h_2 = \tanh\left(W_h \begin{bmatrix} x_2 \\ h_1 \end{bmatrix} + b_h \right)$$



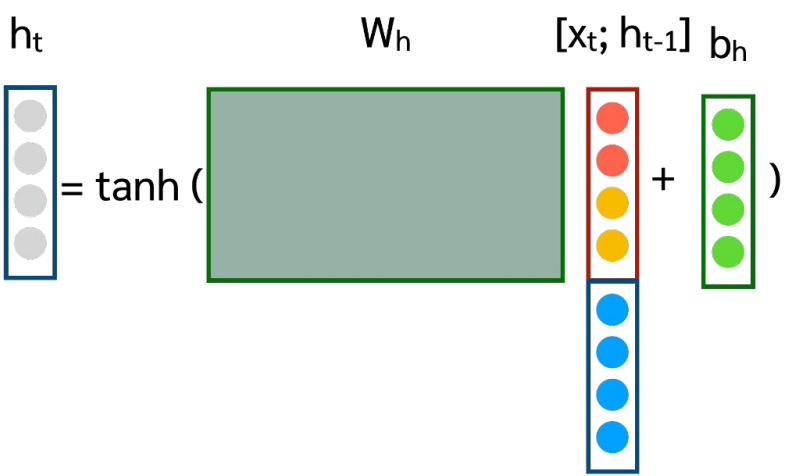
$$y_3 = \text{softmax}(W_y\, h_3 + b_y)$$

$$h_3 = \tanh\left(W_h \begin{bmatrix} x_3 \\ h_2 \end{bmatrix} + b_h \right)$$

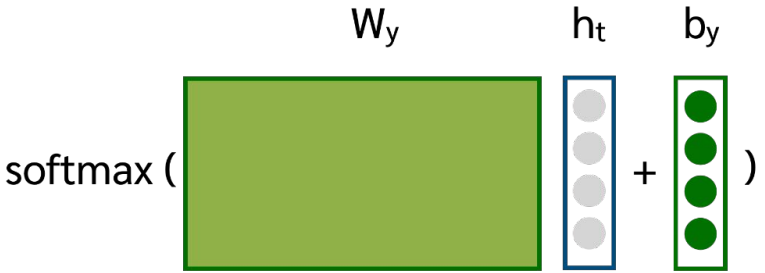

B-PER
I-PER
O
O
B-LOC
Peter
Griffin
lives
in
Quahog

RNN Parameters

$$h_t = \tanh(W_h \cdot [x_t; h_{t-1}]) + b_h$$



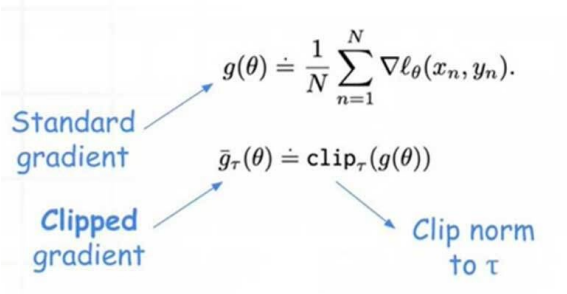
$$y_t = softmax(W_y \cdot h_t + b_y)$$



What are the usual problems? and solution?

- Exploding gradient
- Vanishing gradient

1. Exploding Gradient



Problem:

- During backpropagation, gradients become excessively large.
- Leads to unstable training, dramatic weight updates, and model divergence.

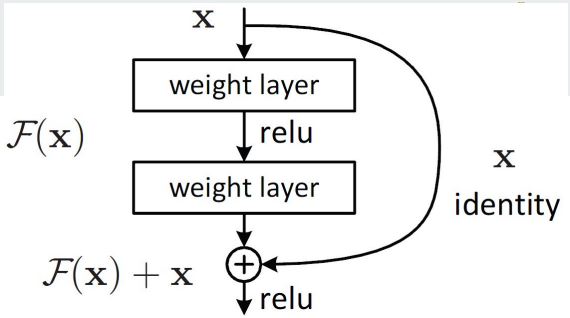
Cause:

- Deep networks multiplying several large values repeatedly during backpropagation.
- Poor initialization or improper learning rates.

Solutions:

- **Gradient clipping:** Set a threshold to limit gradient values, ensuring stability.
- **Reduce Learning Rate:** Smaller learning rates decrease large weight updates.
- **Normalization Techniques:** Employ batch normalization or layer normalization to keep gradients within reasonable bounds.

2.Vanishing Gradient



Problem:

- Gradients become very small as they are propagated backward through the layers.
- Leads to negligible weight updates in early layers, preventing the network from effectively learning.

Cause:

- Multiplication of several small gradients or derivative values (especially in sigmoid or tanh activation functions).
- Deep architectures exacerbate this issue.

Solutions:

- **Use Activation Functions Less Prone to Vanishing:** Replace sigmoid and tanh with ReLU, Leaky ReLU, GELU, ELU, or similar activation functions.
- **Residual Connections (Skip Connections):** Allow earlier layers to receive direct gradients, bypassing intermediate layers (e.g., ResNet architectures).
- **Batch Normalization:** Keeps layer inputs within stable range, improving gradient flow.
- **Gated Architectures:** Use architectures like LSTM or GRU for recurrent networks to prevent gradient decay.

Consider these problems

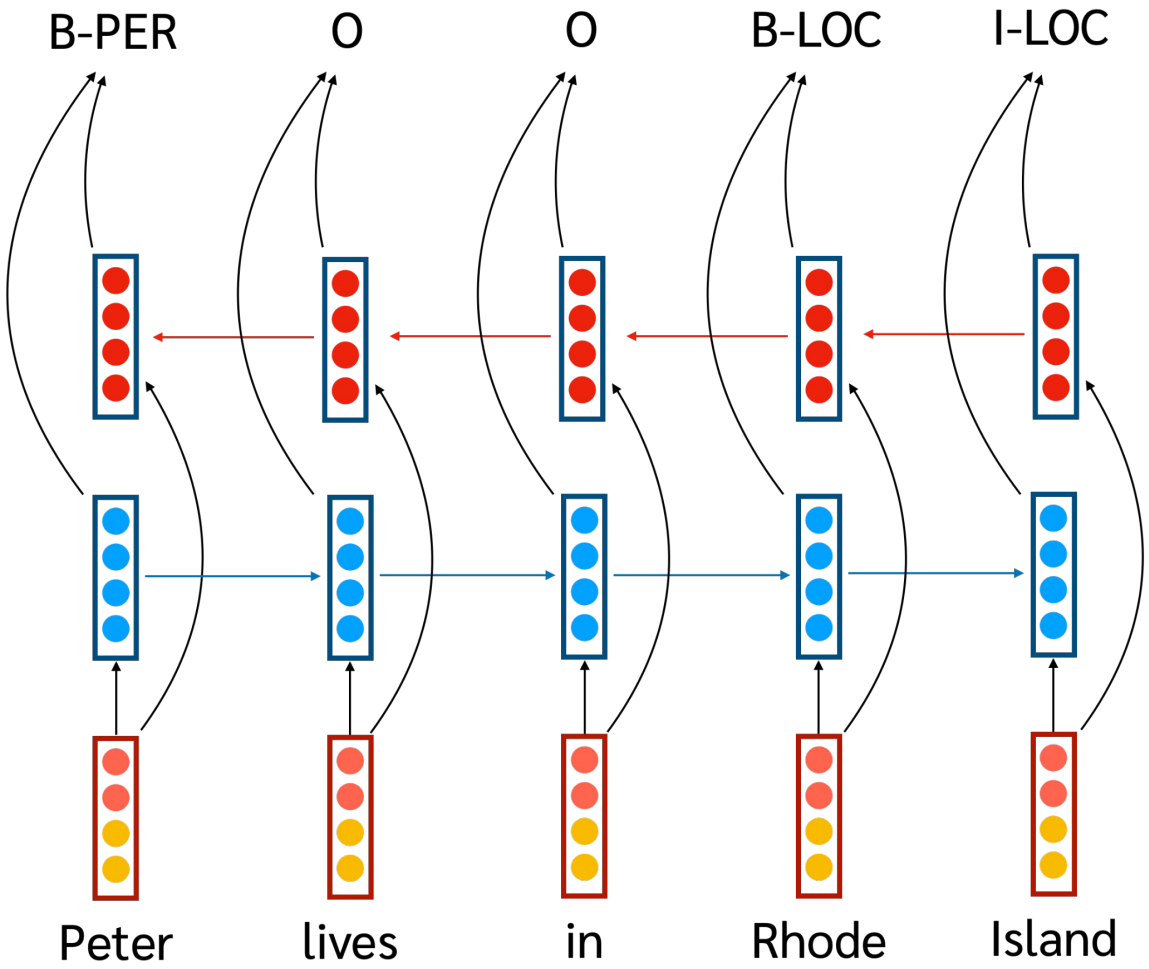
- Millions of tourists flow to Bangkok each year.
- Bangkok is spending more on public green spaces.
- This neighborhood relies on Jones Street for the commute to work.
- My fever got worse until I met Jones.

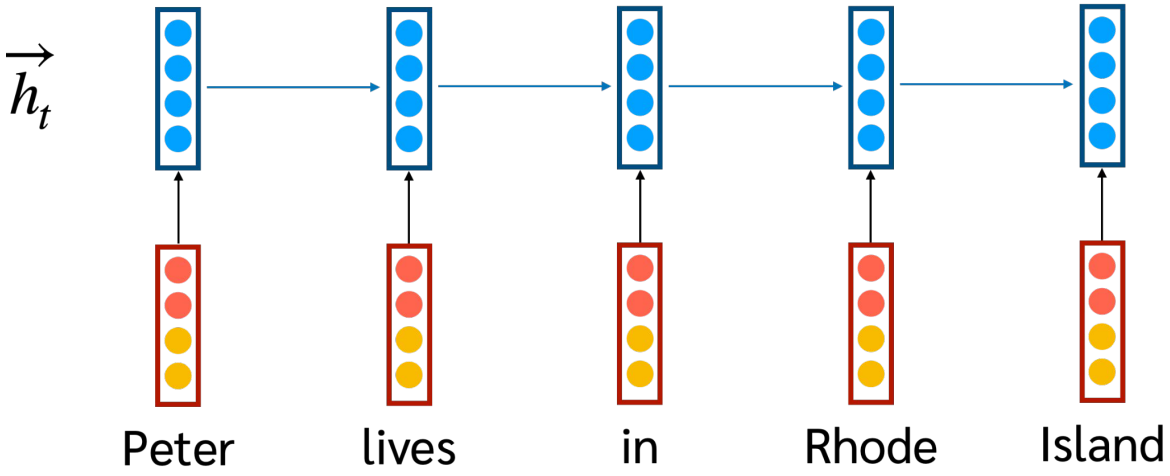
What RNN sees

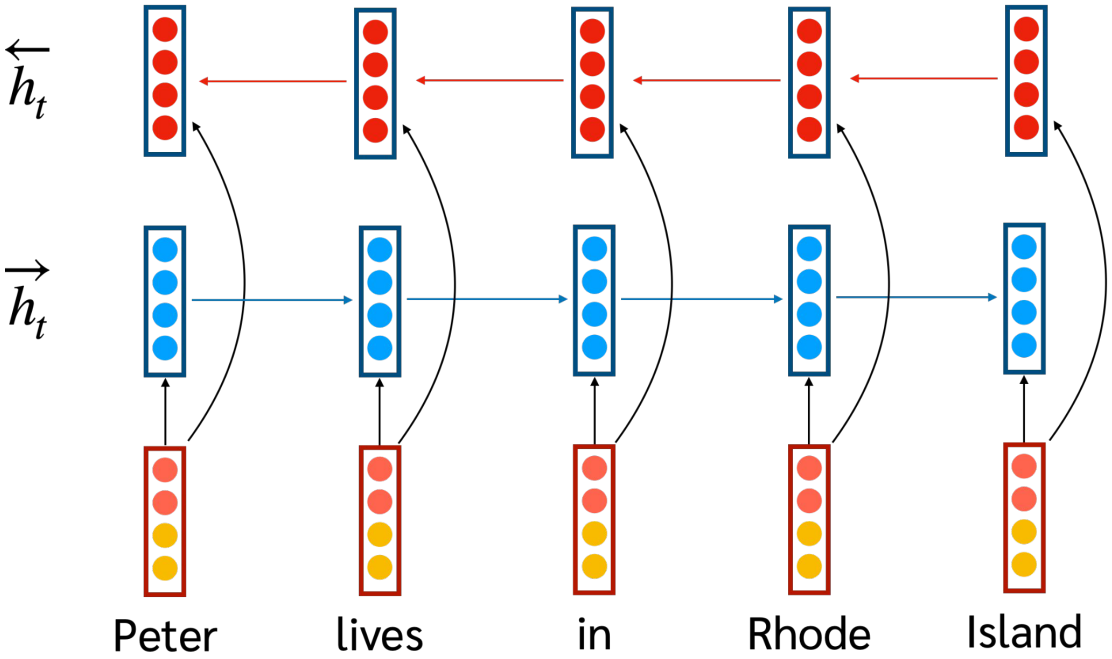
- Millions of tourists flow to Bangkok each year.
- Bangkok is spending more on public green spaces.
- This neighborhood relies on Jones Street for the commute to work.
- My fever got worse until I met Jones.

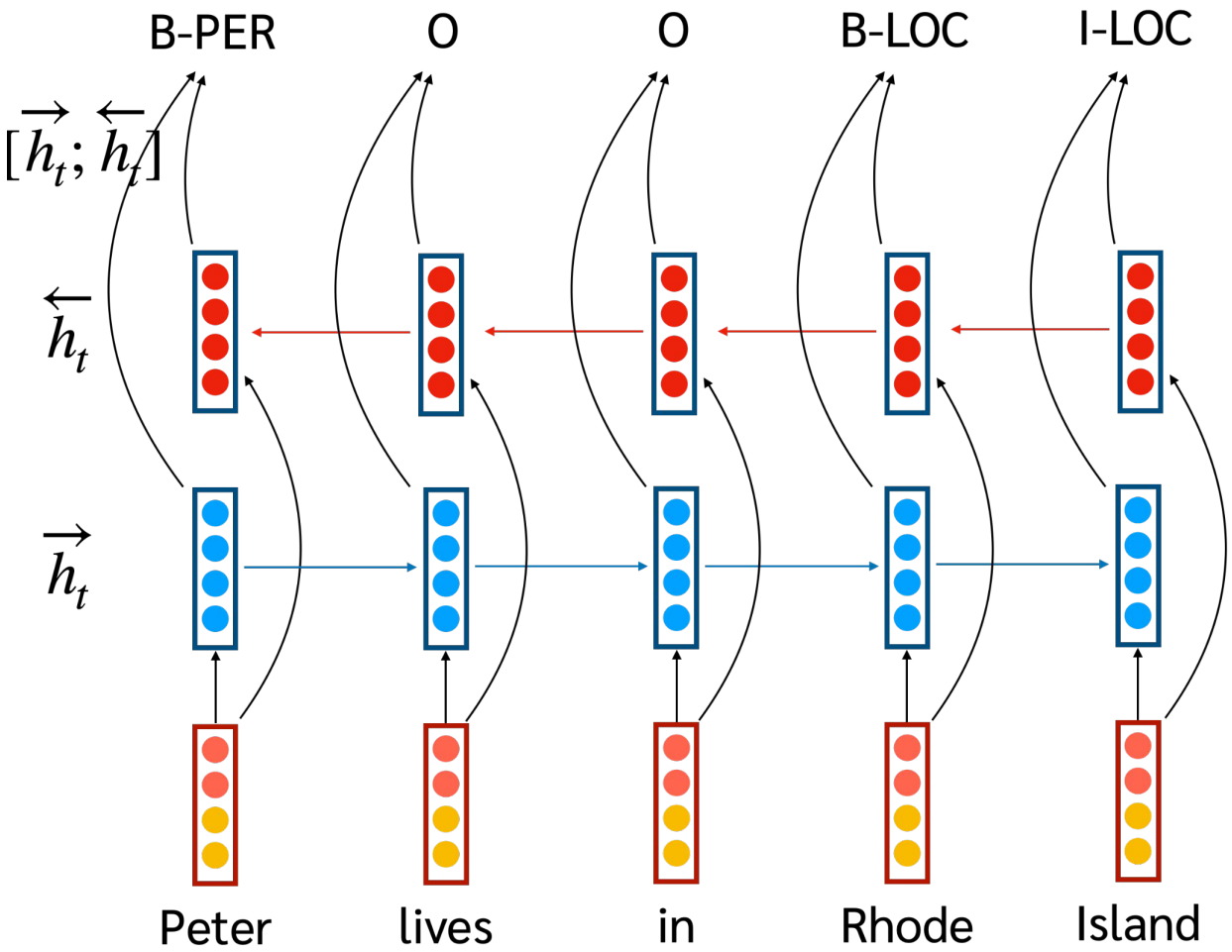
Bi-Directional RNN (Bi-RNN)

- Bi-RNN consists of two RNN's. One processes the input from left to right, and the other processes the input from right to left.
- The feature vector concatenates the hidden activation from the two RNN's.


B-PER
O
O
B-LOC
I-LOC
Peter
lives
in
Rhode
Island

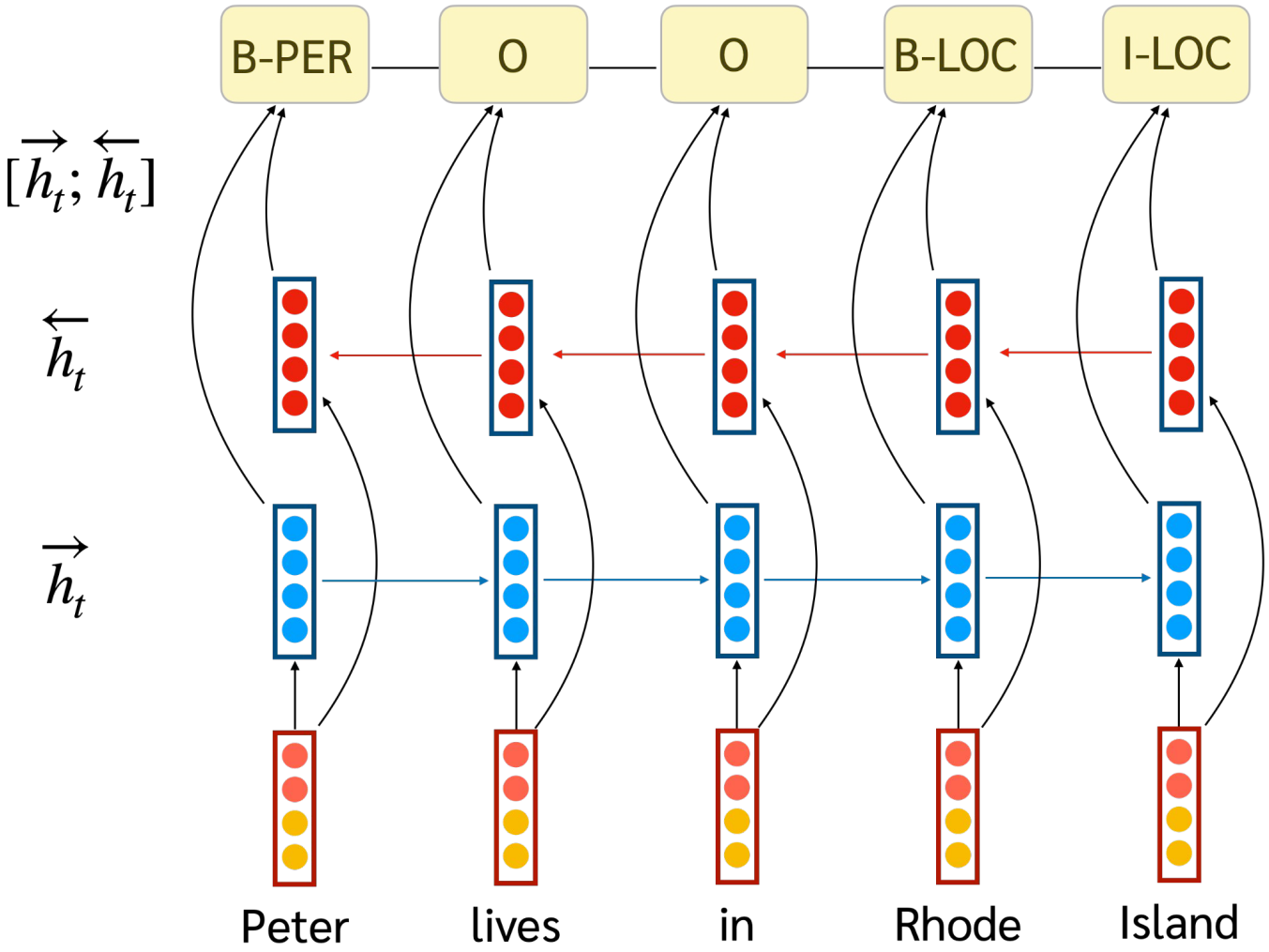

$\overrightarrow{h_t}$
Peter
lives
in
Rhode
Island


$\overleftarrow{h_t}$
$\overrightarrow{h_t}$
Peter
lives
in
Rhode
Island


B-PER
O
O
B-LOC
I-LOC
$[\overrightarrow{h_t}; \overleftarrow{h_t}]$
$\overleftarrow{h_t}$
$\overrightarrow{h_t}$
Peter
lives
in
Rhode
Island

Bi-RNN family

- Vanilla Bi-RNN
- Bi-LSTM (Bi-directional Long Short-term Memory)
- Bi-GRU (Gated Recurrent Unit)


B-PER
O
O
B-LOC
I-LOC
$[\overrightarrow{h_t}; \overleftarrow{h_t}]$
$\overleftarrow{h_t}$
$\overrightarrow{h_t}$
Peter
lives
in
Rhode
Island

Bi-LSTM does quite well

Tagging performance on POS, chunking and NER tasks with only word features.

		POS	CoNLL2000	CoNLL2003
Senna	LSTM	94.63 (-2.66)	90.11 (-2.88)	75.31 (-8.43)
	BI-LSTM	96.04 (-1.36)	93.80 (-0.12)	83.52 (-1.65)
	CRF	94.23 (-3.22)	85.34 (-8.49)	77.41 (-8.72)
	LSTM-CRF	95.62 (-1.92)	93.13 (-1.14)	81.45 (-6.91)
	BI-LSTM-CRF	**96.11** (-1.44)	**94.40** (-0.06)	**84.74** (-4.09)

Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." arXiv preprint arXiv:1508.01991 (2015).

Pretrained embeddings improve performance

		POS	CoNLL2000	CoNLL2003
Random	Conv-CRF (Collobert et al., 2011)	96.37	90.33	81.47
	LSTM	97.10	92.88	79.82
	BI-LSTM	97.30	93.64	81.11
	CRF	97.30	93.69	83.02
	LSTM-CRF	**97.45**	93.80	84.10
	BI-LSTM-CRF	97.43	**94.13**	**84.26**
Senna	Conv-CRF (Collobert et al., 2011)	97.29	94.32	88.67 (89.59)
	LSTM	97.29	92.99	83.74
	BI-LSTM	97.40	93.92	85.17
	CRF	97.45	93.83	86.13
	LSTM-CRF	97.54	94.27	88.36
	BI-LSTM-CRF	**97.55**	**94.46**	**88.83 (90.10)**

Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." arXiv preprint arXiv:1508.01991 (2015).

Almost State-of-the-art POS Tagging

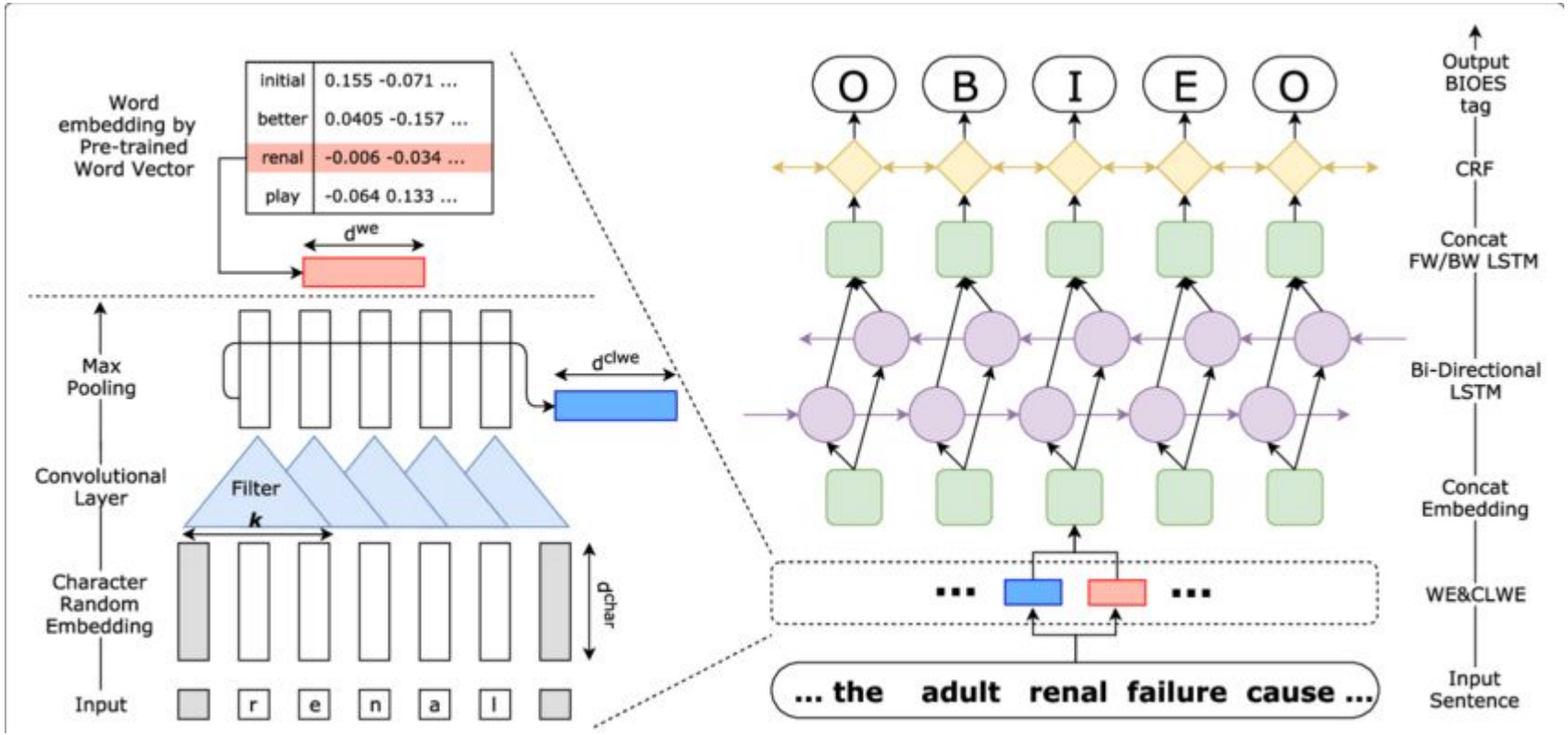
System	accuracy	extra data
Maximum entropy cyclic dependency network (Toutanova et al., 2003)	97.24	No
SVM-based tagger (Gimenez and Marquez, 2004)	97.16	No
Bidirectional perceptron learning (Shen et al., 2007)	97.33	No
Semi-supervised condensed nearest neighbor (Soegaard, 2011)	97.50	Yes
CRFs with structure regularization (Sun, 2014)	97.36	No
Conv network tagger (Collobert et al., 2011)	96.37	No
Conv network tagger (senna) (Collobert et al., 2011)	97.29	Yes
BI-LSTM-CRF (ours)	**97.43**	No
BI-LSTM-CRF (Senna) (ours)	**97.55**	Yes

Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." arXiv preprint arXiv:1508.01991 (2015).

Almost State-of-the-art NER

System	accuracy
Combination of HMM, Maxent etc. (Florian et al., 2003)	88.76
MaxEnt classifier (Chieu., 2003)	88.31
Semi-supervised model combination (Ando and Zhang., 2005)	89.31
Conv-CRF (Collobert et al., 2011)	81.47
Conv-CRF (Senna + Gazetteer) (Collobert et al., 2011)	89.59
CRF with Lexicon Infused Embeddings (Passos et al., 2014)	**90.90**
BI-LSTM-CRF (ours)	84.26
BI-LSTM-CRF (Senna + Gazetteer) (ours)	90.10

Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." arXiv preprint arXiv:1508.01991 (2015).



Encoding a word with character-level CNN

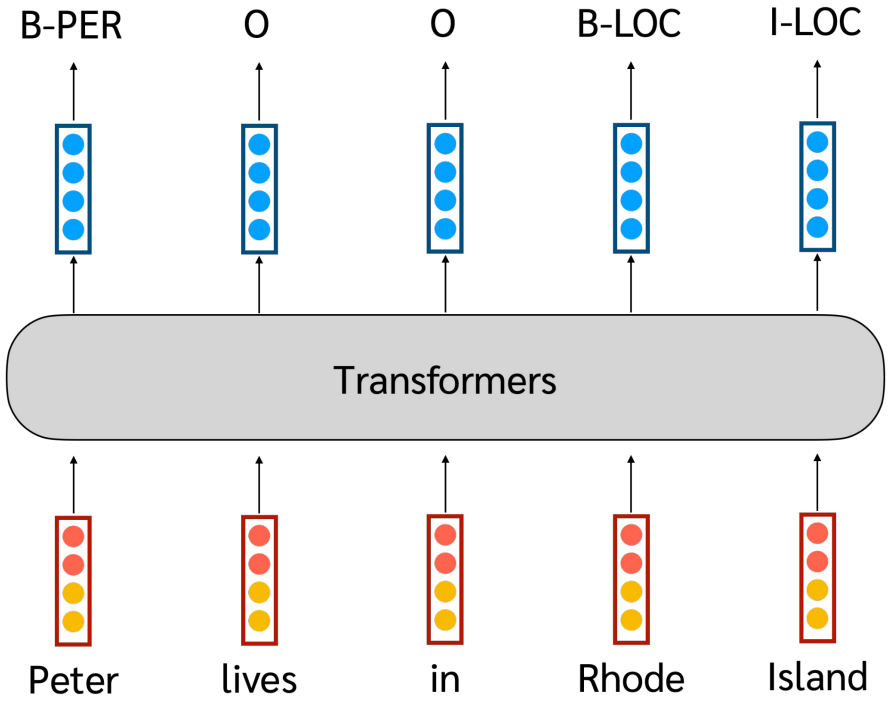
Yoon, Wonjin, et al. "Collabonet: collaboration of deep neural networks for biomedical named entity recognition." *BMC bioinformatics* 20.10 (2019): 55-65.

What we have learned so far

- Embedding features are great for this kind of task. Why?
- Bi-directional models are great for this kind of task. Why?
- Word embeddings derived from sub-word features are great for this task. Why?

Transformers (BERT)

Self-attention is the weighted average of every word in the same sentence. Why is this good?

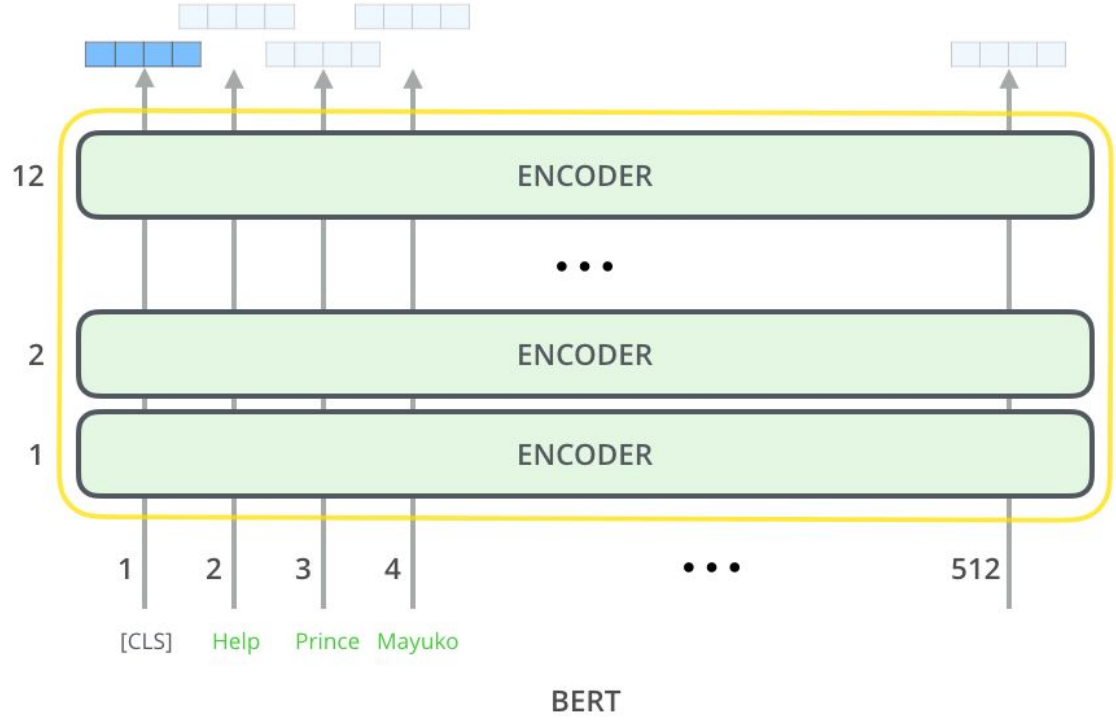


Transformer Tokenizer and Embeddings

Tokenize this: https://platform.openai.com/tokenizer

Overintellectualization can lead to hyperconceptualization and overspecialization

Uncharacteristically, the company's decision to restructure resulted in disestablishmentarianism among its employees.

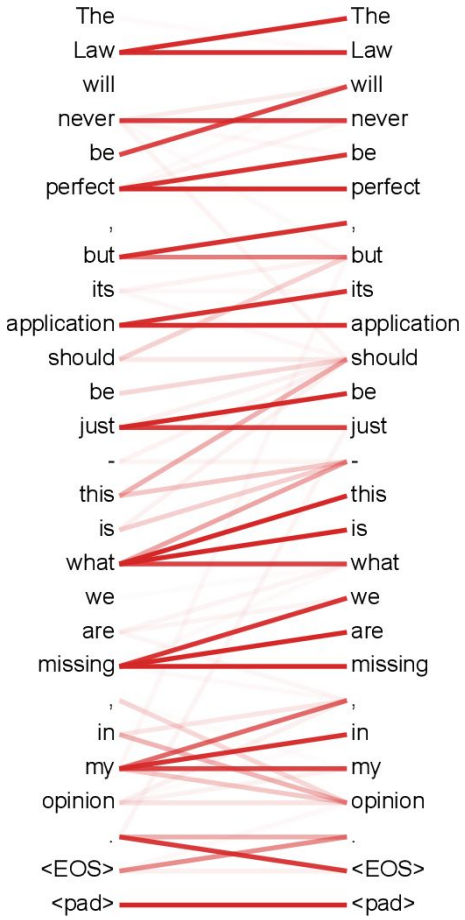


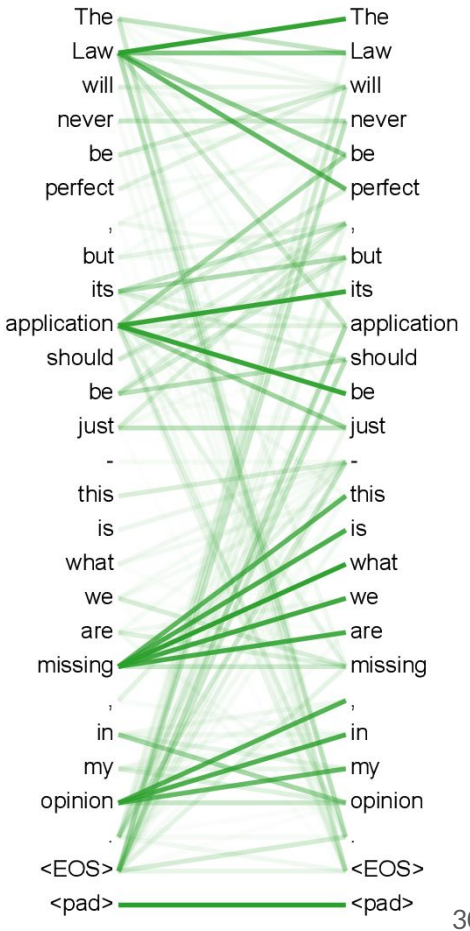
We applied self-attention 12 times.

Self-attention

Each attention network learns to average differently.

What does the red attention learn? And green?





Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Amazing Results on CoNLL 2003

The best BiLSTM-CRF model achieves around 90 F1

System	Dev F1
ELMo (Peters et al., 2018a)	95.7
CVT (Clark et al., 2018)	-
CSE (Akbik et al., 2018)	-
Fine-tuning approach	
$\text{BERT}_{\text{LARGE}}$	96.6
$\text{BERT}_{\text{BASE}}$	96.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)	
Embeddings	91.0
Second-to-Last Hidden	95.6
Last Hidden	94.9
Weighted Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1
Weighted Sum All 12 Layers	95.5

Conclusion

- The world has come so far as to do sequence tagging.
- CRF is a keyword-based model that performs decently well.
- The RNN family model performs well when coupled with other discrete features, but it requires much tuning for it to work well.
- Transformer-based model performs the best at the cost of computational power at runtime.

NER Tutorial

NER data in 'CoNLL' format

https://www.kaggle.com/datasets/alaakhaled/conll003-englishversion

one line per token

one column per info
(tab separated columns)

West	NNP	B-NP	B-MISC
Indian	NNP	I-NP	I-MISC
all-rounder	NN	I-NP	O
Phil	NNP	I-NP	B-PER
Simmons	NNP	I-NP	I-PER
took	VBD	B-VP	O
four	CD	B-NP	O
for	IN	B-PP	O
38	CD	B-NP	O
on	IN	B-PP	O
Friday	NNP	B-NP	O
as	IN	B-PP	O
Leicestershire	NNP	B-NP	B-ORG
beat	VBD	B-VP	O
Somerset	NNP	B-NP	B-ORG

Steps

1. Read the text data into a list of [lists of word strings]
   and the label into a list of [lists of label strings]
2. Read the pretrained word embeddings and prepare vocabulary
   a. reserve index 0 for padding
   b. reserve index 1 for <UNK> (this is important. why?)
3. Convert label strings to label index (use dictionary or LabelEncoder)
4. Make all sequences the same length (padding if too short. truncating if too long)
5. Split train-dev-test sets