Neural Machine Translation

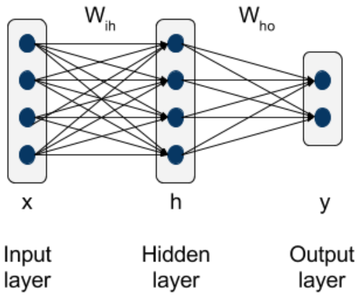
Encoder-Decoder

Iacer Calixto

Institute for Logic, Language and Computation
University of Amsterdam

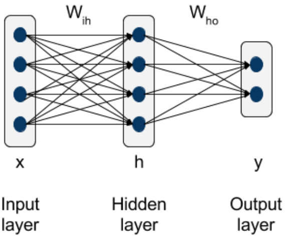
May 18, 2018

Artificial Neural Networks [1]



Let $\boldsymbol{x} \in \mathbb{R}^4, \boldsymbol{h} \in \mathbb{R}^4, \boldsymbol{y} \in \mathbb{R}^2$.

Artificial Neural Networks [2]



Let $\boldsymbol{x} \in \mathbb{R}^4, \boldsymbol{h} \in \mathbb{R}^4, \boldsymbol{y} \in \mathbb{R}^2$,

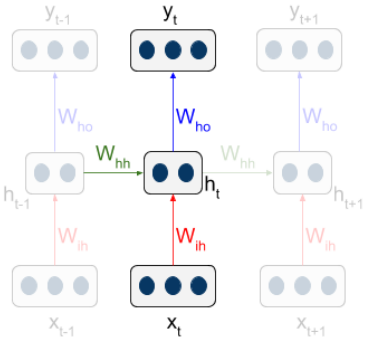
$\boldsymbol{W}_{ih} \in \mathbb{R}^{4\times4}$ and $\boldsymbol{b}_{ih} \in \mathbb{R}^4$, and

$\boldsymbol{W}_{ho} \in \mathbb{R}^{4\times2}$ and $\boldsymbol{b}_{ho} \in \mathbb{R}^2$.

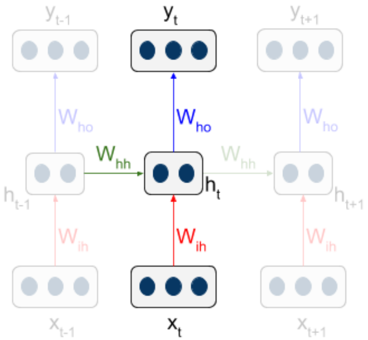
$$\boldsymbol{h} = \mathrm{f}(\boldsymbol{x}^T \boldsymbol{W}_{ih} + \boldsymbol{b}_{ih}),$$

$$\boldsymbol{y} = \mathrm{g}(\boldsymbol{h}^T \boldsymbol{W}_{ho} + \boldsymbol{b}_{ho}).$$

Recurrent Neural Networks[1]



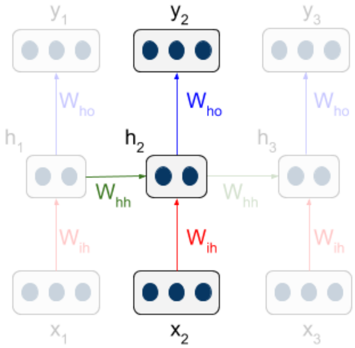
Recurrent Neural Networks[2]



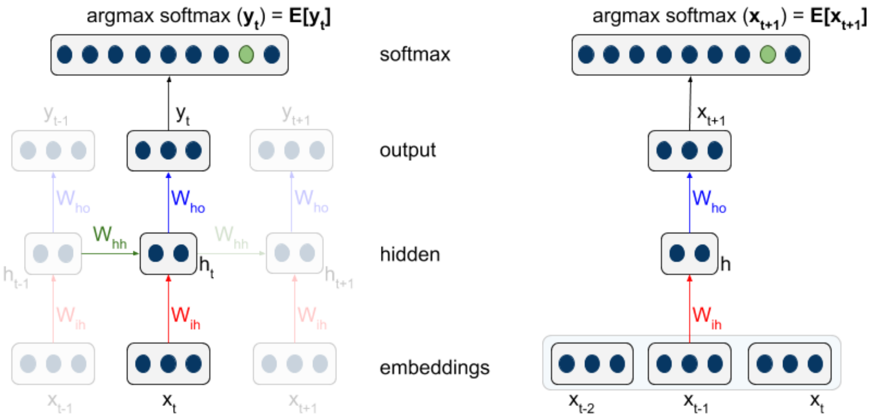
$$\boldsymbol{h}_t = \mathrm{f}(\boldsymbol{W}_{ih}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{ih}),$$
$$\boldsymbol{y}_t = \mathrm{g}(\boldsymbol{W}_{ho}\boldsymbol{h}_t + \boldsymbol{b}_{ho}).$$

Recurrent Neural Networks[3]

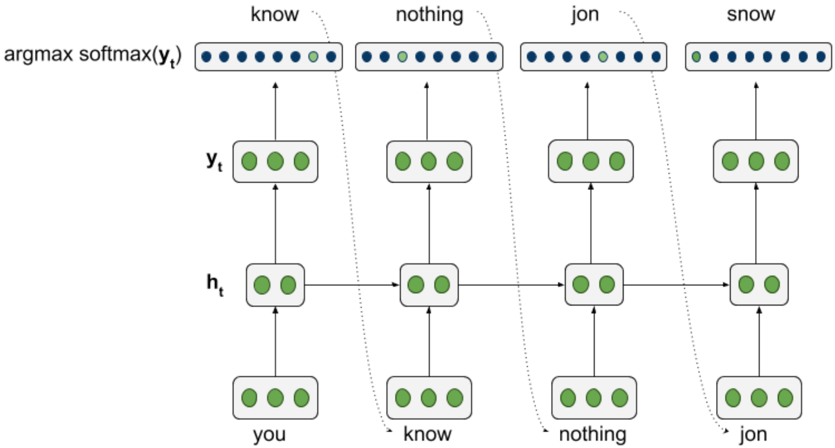
- For a **sequence of input vectors** $x = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$, an RNN will compute a **sequence of hidden states** $H = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3\}$, and optionally a **sequence of output vectors** $y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3\}$.



RNN vs. FFNN

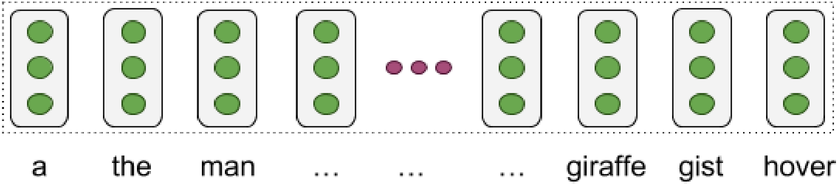

argmax softmax (y_t) = E[y_t]
softmax
y_t-1
y_t
y_t+1
output
W_ho
W_ho
W_ho
W_hh
W_hh
hidden
h_t-1
h_t
h_t+1
W_ih
W_ih
W_ih
embeddings
x_t-1
x_t
x_t+1
argmax softmax (x_t+1) = E[x_t+1]
x_t+1
W_ho
h
W_ih
x_t-2
x_t-1
x_t

RNN Language Model



Word Embeddings: what are they?

Word embedding matrix:



Tipically, an embedding matrix is denoted by $\boldsymbol{W}$ or $\boldsymbol{E}$.
$\boldsymbol{E}_x$: source-language embeddings;
$\boldsymbol{E}_y$: target-language embeddings.

$$\boldsymbol{E} \in \mathbb{R}^{|V| \times d},$$

where V is the vocabulary and d is the word embedding dimensionality.

Word Embeddings: where do they come from?

Random initialisation (when enough training data is available)
E.g. Sample from a uniform distribution [-0.1,+0.1];

Supervised pre-training
Train the embeddings first in a task for which there is abundant data;

Unsupervised pre-training
Create your own supervised task from raw text (e.g. word2vec);

Word Embeddings: word2vec (Mikolov et al., 2014)

Continuous Bag-Of-Words Model (CBOW)

The model predicts the current word given the surrounding words. Supervision is obtained by iterating a corpus and using a fixed window to gather surrounding words.

Example:

$\cdots$ finished . the cat jumped like crazy and the giraffe $\cdots$

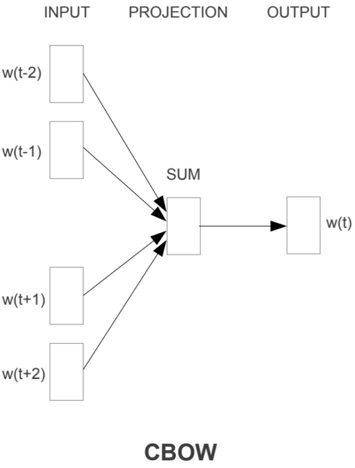
$$\text{Input } X = \{x_1, x_2, \_, x_4, x_5\}$$
$$\text{Output } Y = \{x_3\}$$

Word Embeddings: word2vec (Mikolov et al., 2014)

Example:

$\cdots$ the cat jumped like crazy $\cdots$

Input $X = \{x_1, x_2, \_, x_4, x_5\}$
Output $Y = \{x_3\}$



Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997)

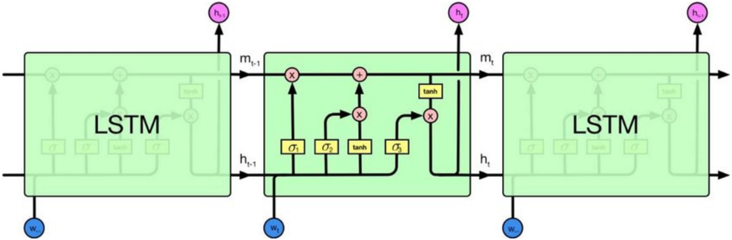


Image credits: Ma, Xiang, Du, and Fan. (2018).

$$\boldsymbol{i_t} = \sigma(W_i \boldsymbol{x_t} + U_i \boldsymbol{h_{t-1}})$$
$$\boldsymbol{f_t} = \sigma(W_f \boldsymbol{x_t} + U_f \boldsymbol{h_{t-1}})$$
$$\boldsymbol{o_t} = \sigma(W_o \boldsymbol{x_t} + U_o \boldsymbol{h_{t-1}})$$
$$\boldsymbol{a_t} = \tanh(W_c \boldsymbol{x_t} + U_c \boldsymbol{h_{t-1}})$$

$$\boldsymbol{c_t} = \boldsymbol{i_t} \odot \boldsymbol{a_t} + \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}}$$
$$\boldsymbol{h_t} = \boldsymbol{o_t} \odot \tanh(\boldsymbol{c_t})$$

Some different roles RNNs take

Given a sequence of inputs $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$, in short $\boldsymbol{x}_{1:n}$:

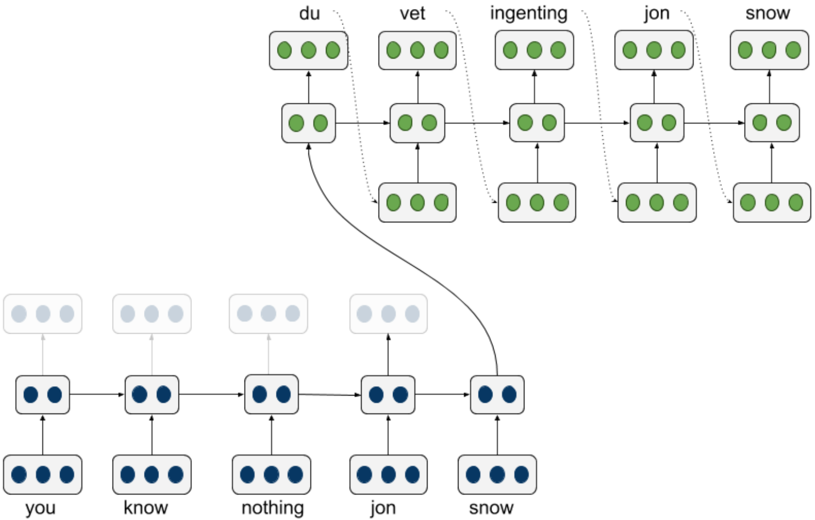
- Encoder: compute a sequence of hidden states $\boldsymbol{h}_{1:n}$, or perhaps we just need to encode the entire sequence X into a fixed-size vector $\boldsymbol{h}_n$;
- Acceptor: accept/reject X;
  - spam detection, sentiment classification;
- Transducer: compute a sequence of outputs for each $\boldsymbol{x}_i$;
  - part-of-speech tagging, language modelling;
- Encoder-Decoder: encode X and use the last hidden state $\boldsymbol{h}_n$ to initialise another RNN that generates a sequence of output words $y_{1:m}$;
  - machine translation, text summarisation;

Encoder–Decoder or seq2seq (Cho et al., 2014; Sutskever et al., 2014)

Components:

- Encoder: projects the source-language sentence X into a fixed-dimensional feature vector $\boldsymbol{h}$;
- Decoder: generates the target-language translation Y of X from $\boldsymbol{h}$;
- Typically, encoder and decoder are both LSTM networks.

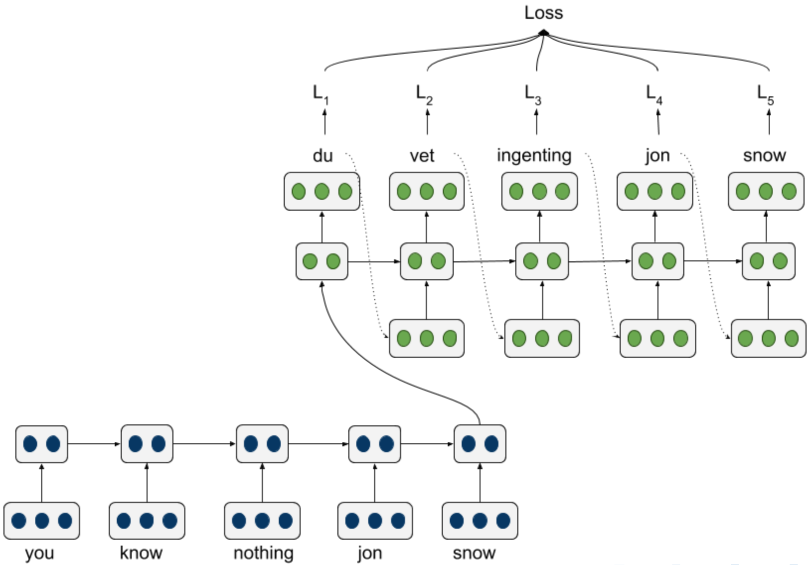
Encoder–Decoder



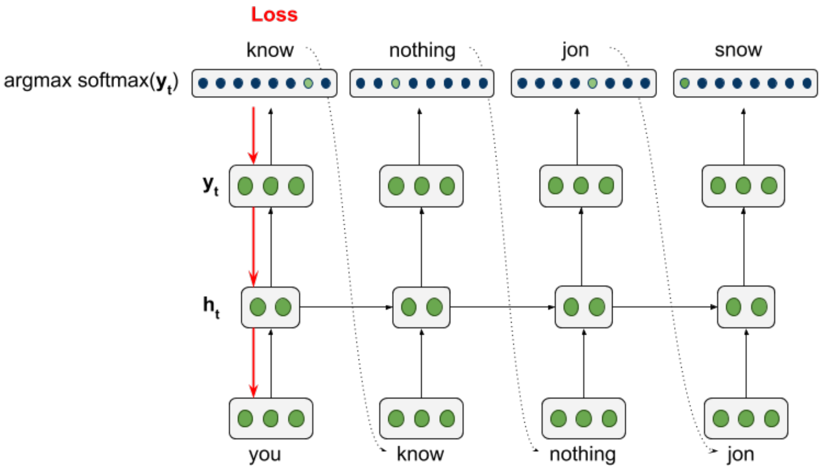
Encoder–Decoder: step-by-step

- Word embeddings
  - source: $\boldsymbol{E}_x$["you"], $\boldsymbol{E}_x$["know"], $\boldsymbol{E}_x$["nothing"], $\boldsymbol{E}_x$["john"], $\boldsymbol{E}_x$["snow"]
  - target: $\boldsymbol{E}_y$["du"], $\boldsymbol{E}_y$["vet"], $\boldsymbol{E}_y$["ingenting"], $\boldsymbol{E}_y$["john"], $\boldsymbol{E}_y$["snow"]
  - source a.k.a.: $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_5\}$
  - target a.k.a.: $Y = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_5\}$
  - In short: $X = \boldsymbol{x}_{1:5}$ and $Y = \boldsymbol{y}_{1:5}$.
- Encoder
  - $\boldsymbol{h}_0 = \vec{0}$;
  - $\boldsymbol{h}_{1:5} = \text{LSTM}_x(\boldsymbol{x}_{1:5})$;
- Decoder
  - $\boldsymbol{s}_0 = \text{mean}(\boldsymbol{h}_{1:5})$, or $\boldsymbol{s}_0 = \boldsymbol{h}_5$;
  - $\boldsymbol{s}_{1:5} = \text{LSTM}_y(\boldsymbol{y}_{1:5})$
- Readout: $\hat{\boldsymbol{y}}_{1:5} = \text{argmax softmax}(\boldsymbol{s}_{1:5})$
- Loss: $\mathcal{L}(\hat{Y}, Y) = \sum_i \mathcal{L}(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i)$

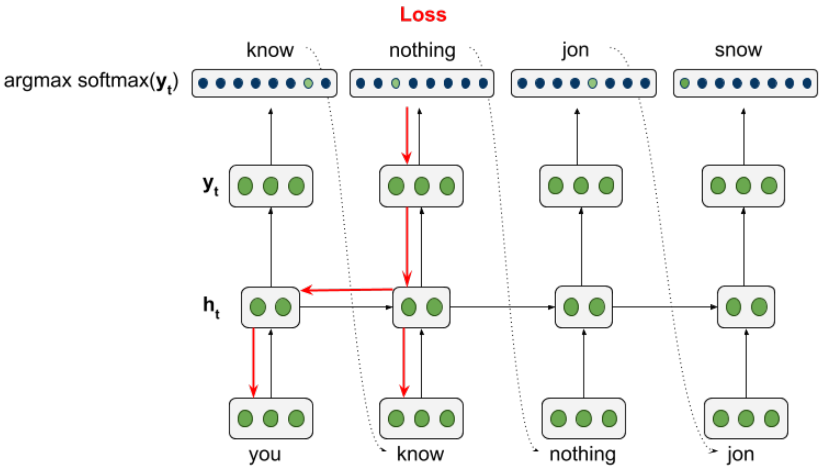
Encoder–Decoder: step-by-step



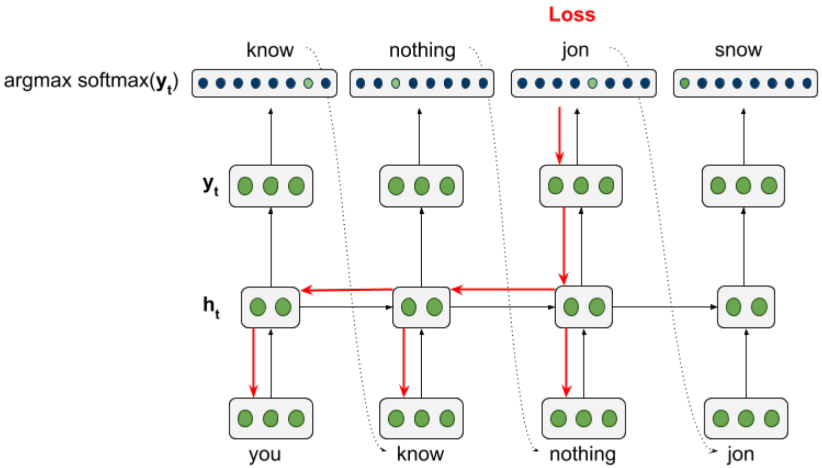
An Idea on Backpropagation Through Time



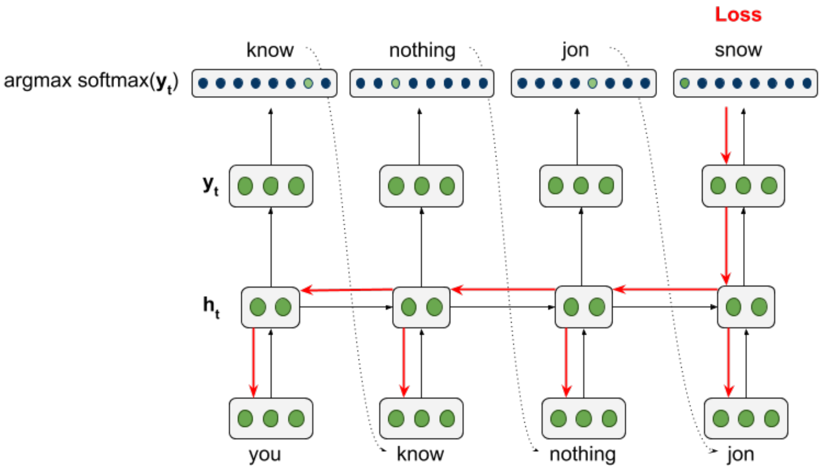
An Idea on Backpropagation Through Time



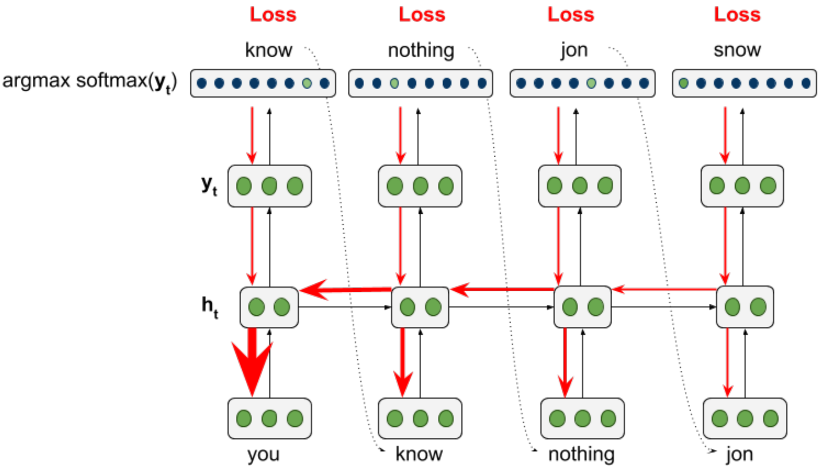
An Idea on Backpropagation Through Time



An Idea on Backpropagation Through Time



An Idea on Backpropagation Through Time



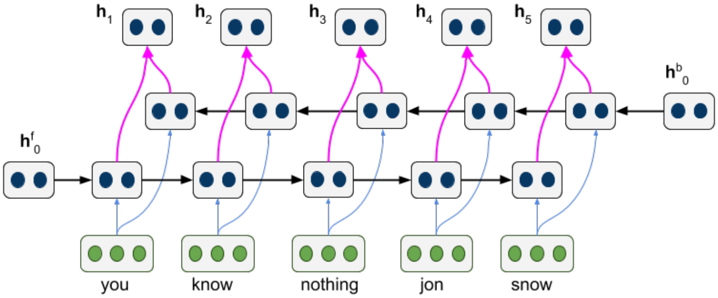
Vanishing and Exploding Gradients

Vanilla RNNs are difficult to train because they suffer from the "vanishing gradients" problem.

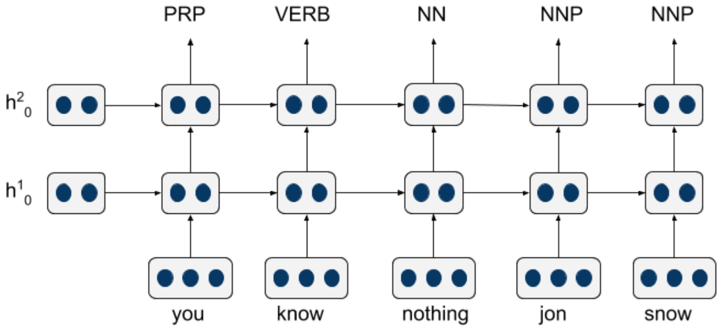
During training with back-propagation, gradients quickly become small as the length of the RNN grows because of the chain rule.

In more rare situations, it is also possible that gradients explode.

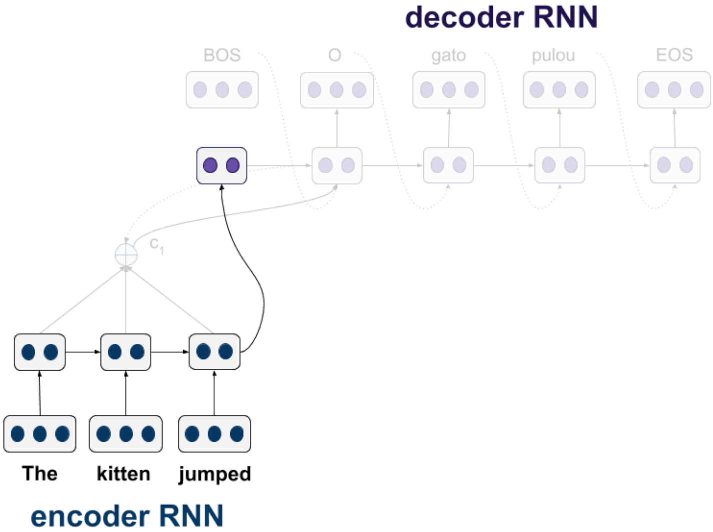
Bidirectional RNNs



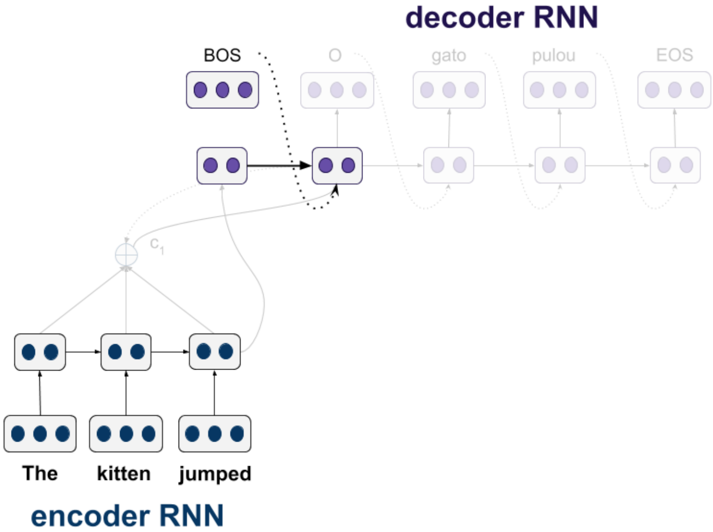
Multilayer RNNs



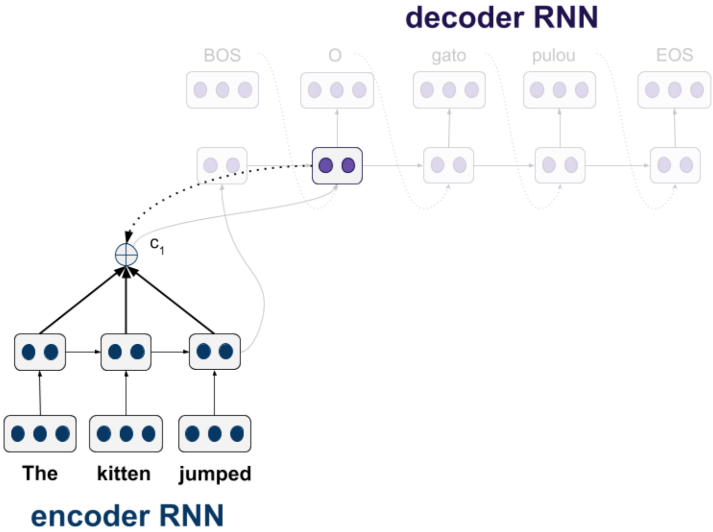
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



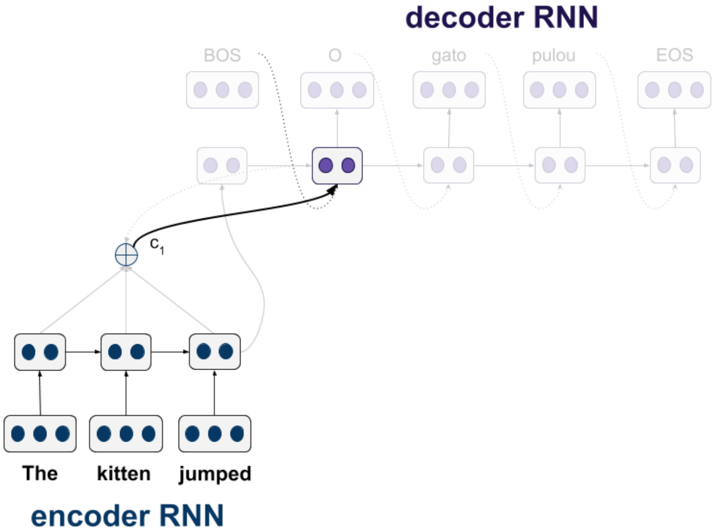
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



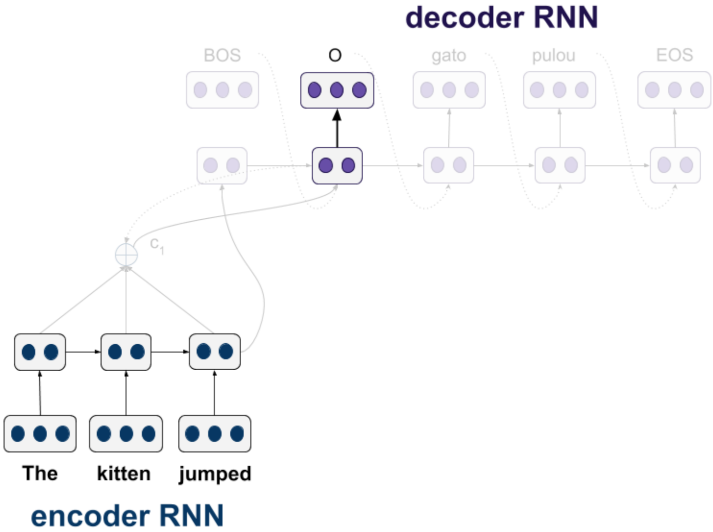
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



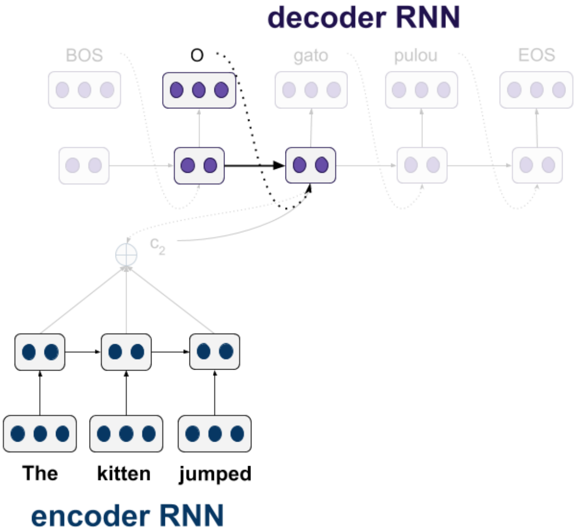
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



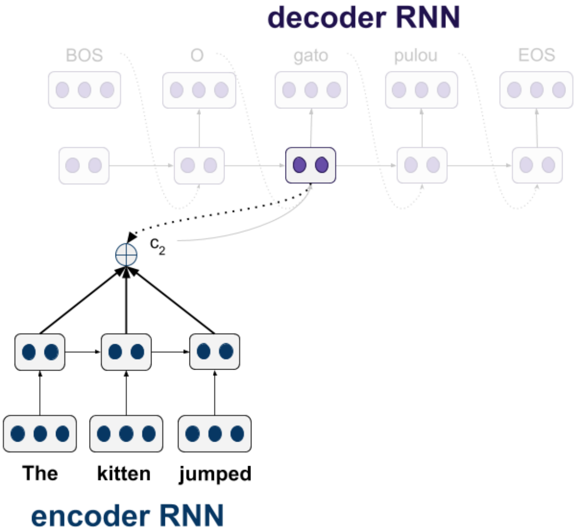
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



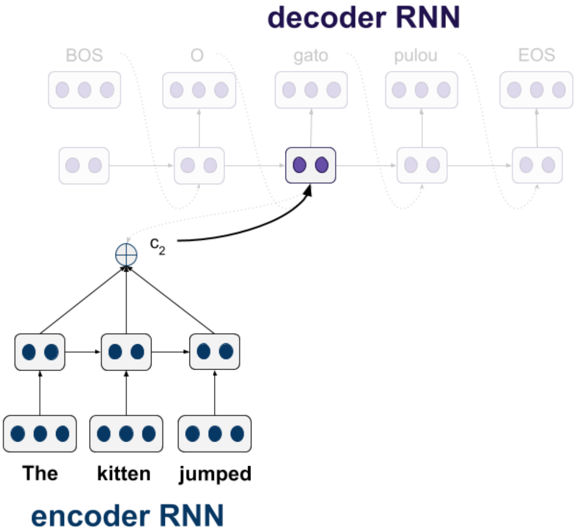
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



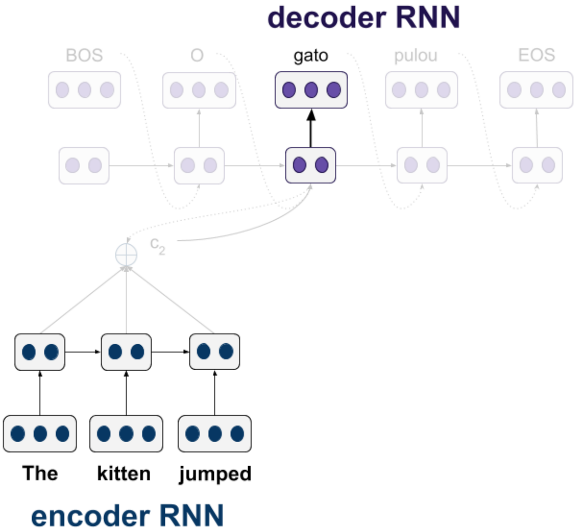
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



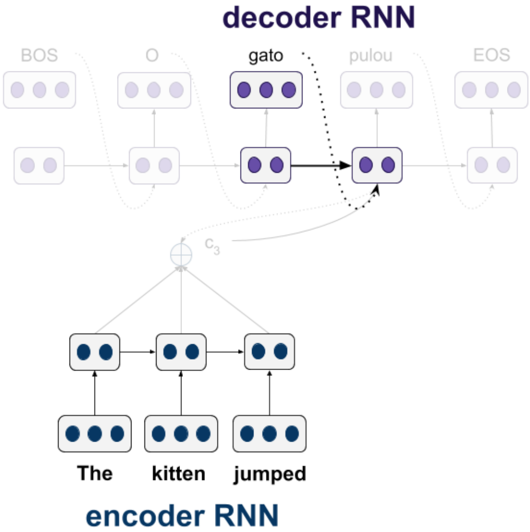
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



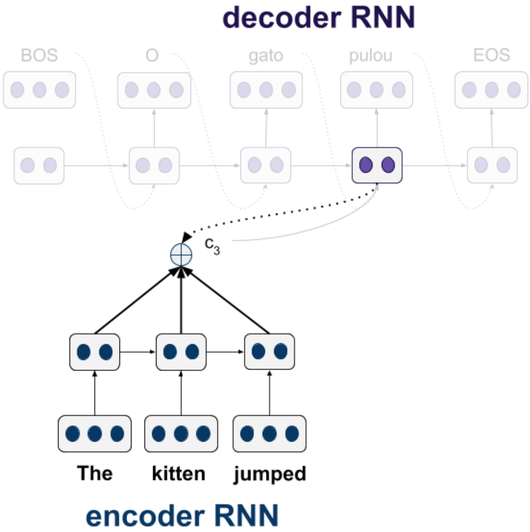
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



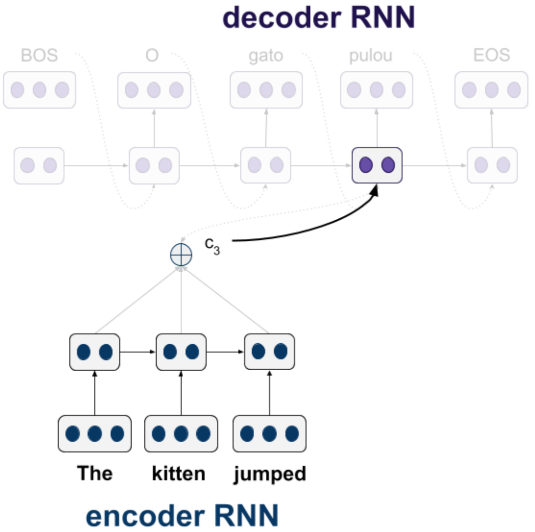
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



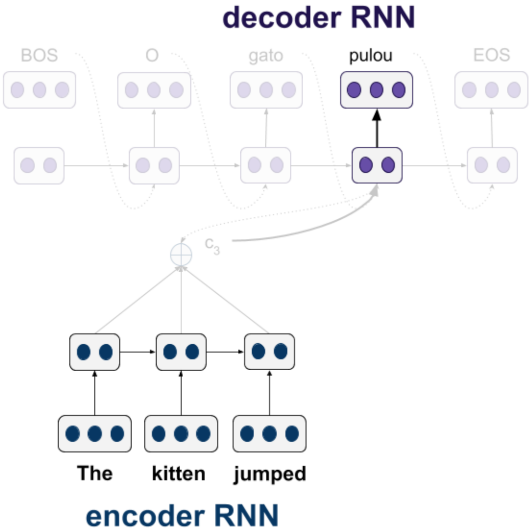
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



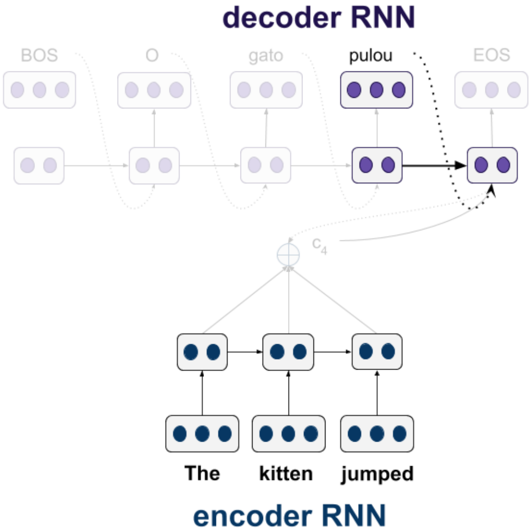
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



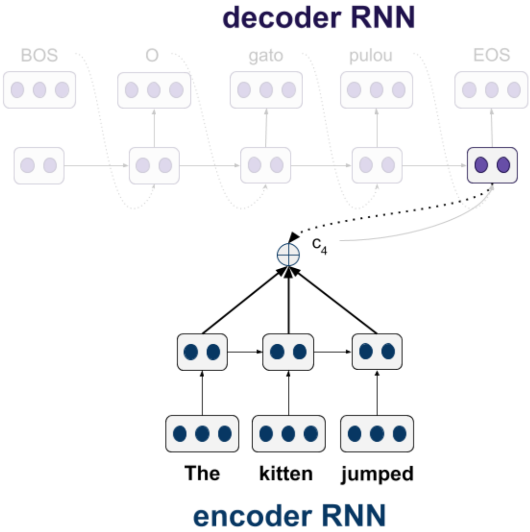
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



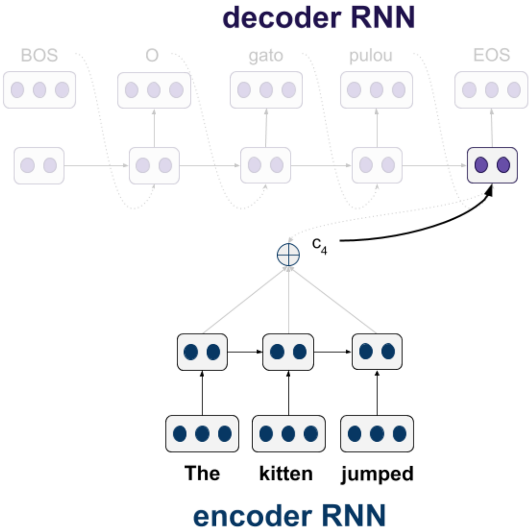
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



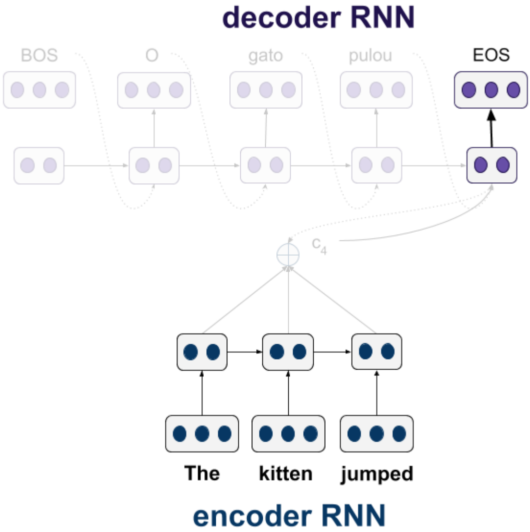
Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



Encoder-Decoder with Attention (Bahdanau et al., 2014; Luong et al., 2015)



Unknown Words

Softmax is a very expensive operation.

That means we must limit the target vocabulary, e.g. most frequent 50k words.

Any other words (i.e., out-of-vocabulary words) are now translated as `UNK`.

What can we do about this?

Using byte-pair encodings (Sennrich et al., 2016)

Start with a vocabulary of characters only.

Repeat: replace each most frequent pair ('A', 'B') with a new symbol 'AB'.

Dictionary:

```
5 l o w </w>
2 l o w e r </w>
6 n e w e s t </w>
3 w i d e s t </w>
```

Vocabulary:

```
l o w e r n s t i d
```

Using byte-pair encodings

Start with a vocabulary of characters only.

Repeat: replace each most frequent pair ('A', 'B') with a new symbol 'AB'.

Dictionary:
5 l o w </w>
2 l o w e r </w>
6 n e w **e** **s** t </w>
3 w i d **e** **s** t </w>

Vocabulary:
l o w e r n s t i d **es**

Add pair ('e','s') with a frequency of **9**.

Using byte-pair encodings

Start with a vocabulary of characters only.

Repeat: replace each most frequent pair ('A', 'B') with a new symbol 'AB'.

Dictionary:

```
5 l o w </w>
2 l o w e r </w>
6 n e w e s t </w>
3 w i d e s t </w>
```

Vocabulary:

`l o w e r n s t i d es` **`est`**

Add pair ('es', 't') with a frequency of **9**.

Using byte-pair encodings

Start with a vocabulary of characters only.

Repeat: replace each most frequent pair ('A', 'B') with a new symbol 'AB'.

Dictionary:

`5` **`l o`** `w </w>`

`2` **`l o`** `w e r </w>`

`6 n e w e s t </w>`

`3 w i d e s t </w>`

Vocabulary:

`l o w e r n s t i d es est`

`lo`

Add pair ('l', 'o') with a frequency **7**.

References

Bahdanau, Cho, and Bengio (2014). Neural Machine Translation by Jointly Learning to Align and Translate. Arxiv pre-print: 1409.0473.

Hochreiter and Schmidhuber (1997). Long Short-Term Memory. Journal Neural Computation. Volume 9, Issue 8, November 1997. p.1735-1780.

Luong, Pham,and Manning (2015). Effective Approaches to Attention-based Neural Machine Translation. In: EMNLP 2015.

Ma, Xiang, Du, and Fan. (2018). Effects of user-provided photos on hotel review helpfulness: An analytical approach with deep learning. International Journal of Hospitality Management. 71. 120-131.

Mikolov, Chen, Corrado, and Dean (2014). Efficient Estimation of Word Representations in Vector Space. Arxiv pre-print: 1301.3781.

Sennrich, Haddow, and Birch (2016). Neural Machine Translation of Rare Words with Subword Units. In: ACL 2016.

Sutskever, Vinyals, and Le (2014). Sequence to Sequence Learning with Neural Networks. In: NIPS 2014.