Neural Machine Translation

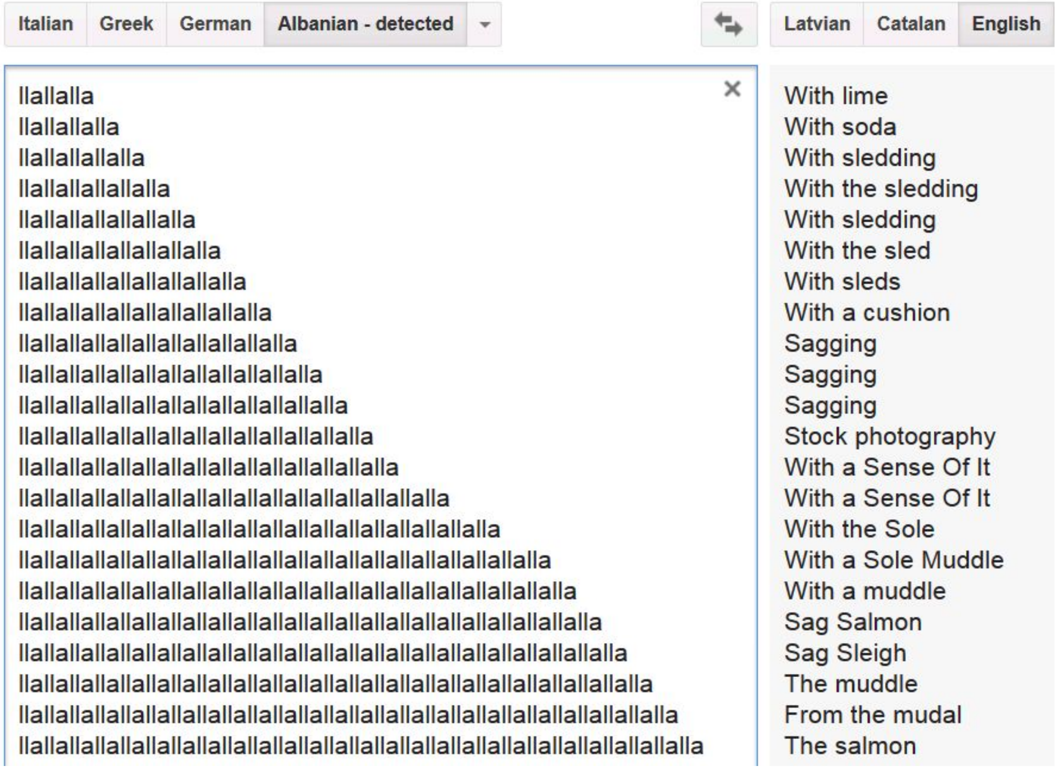
Joost Bastings

http://joost.ninja

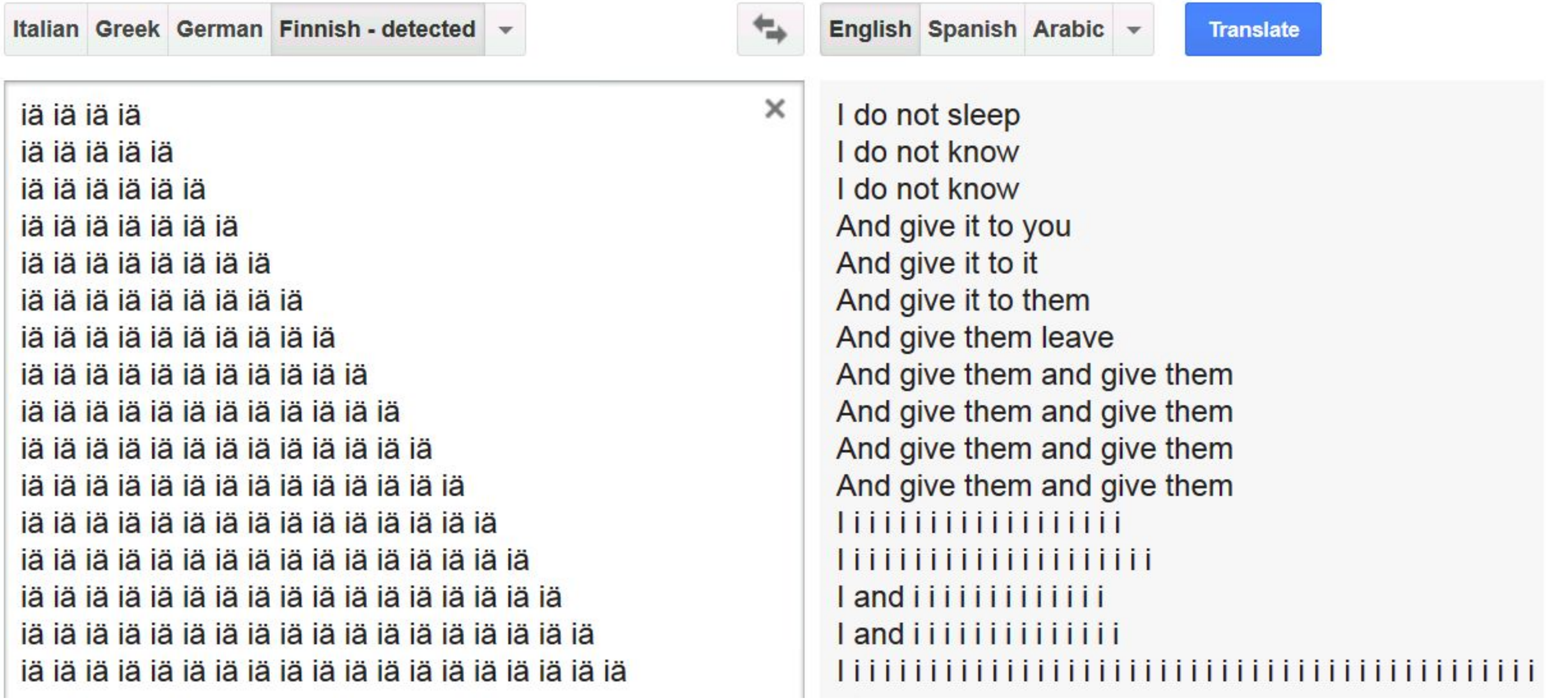
Google Translate

Demo: Sampling from a neural conditional language model

Google Translate



Google Translate



today

recap: RNNs

encoder-decoder

attention models

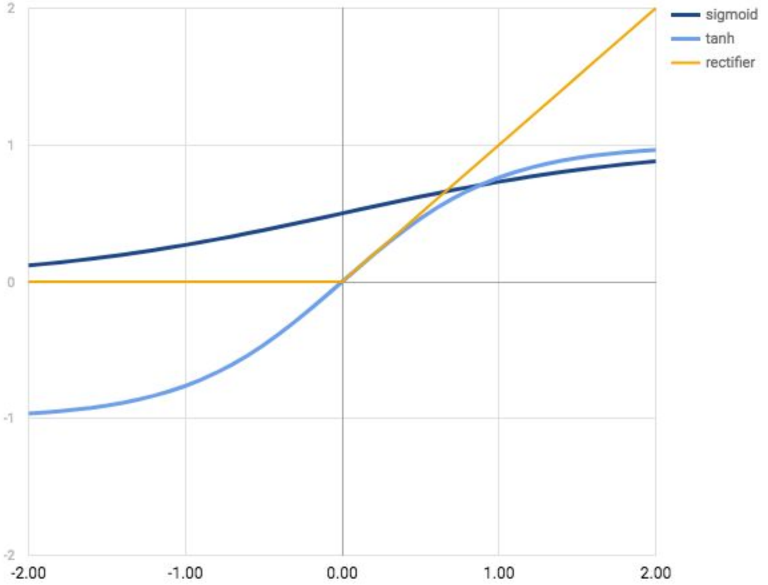
dealing with unknown words

Recap

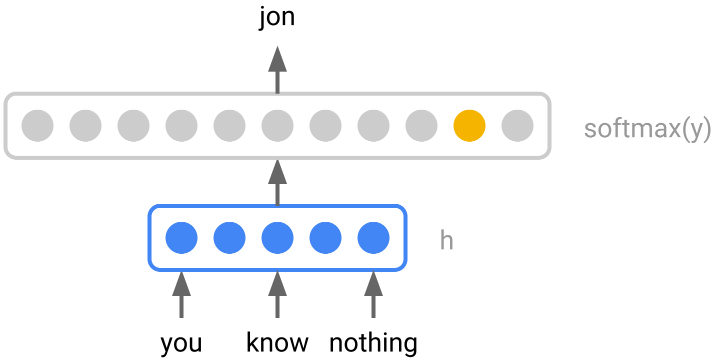
Recap: Matrix Multiplication

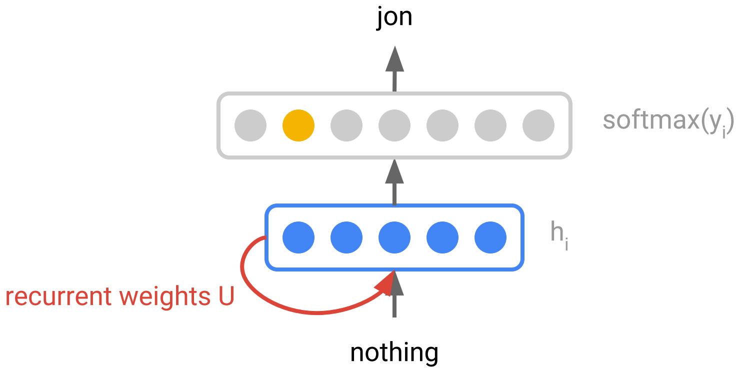
$$\underset{2\text{x}3}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}} \times \underset{3\text{x}2}{\begin{bmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{bmatrix}} =$$

Recap: Activation functions



FFNN vs. RNN





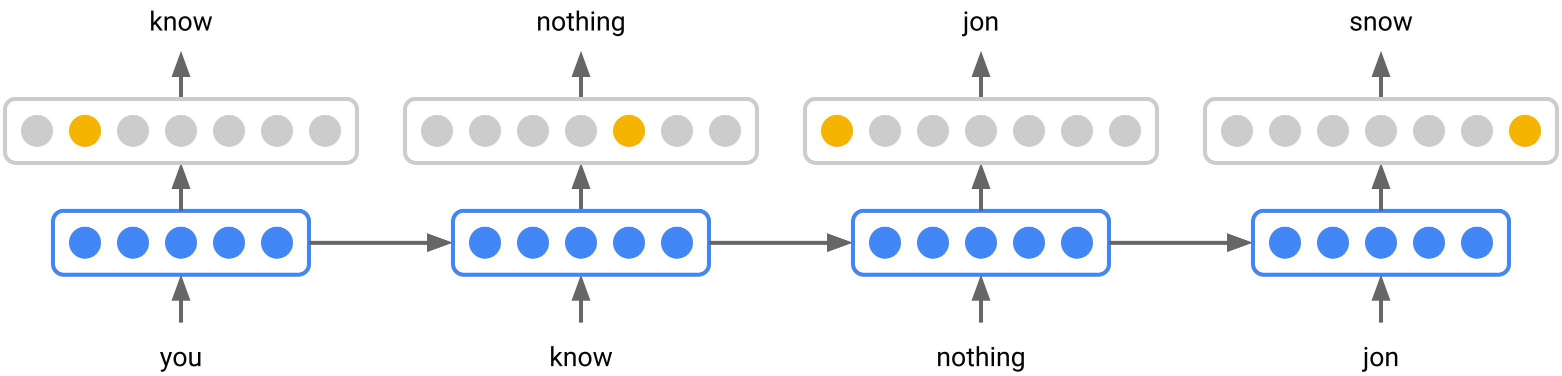
$$\mathbf{h} = \phi(W\mathbf{x} + \mathbf{b})$$

$$\mathbf{y} = W' \, \mathbf{h} + \mathbf{b}'$$

$$\mathbf{h}_i = \phi(W\mathbf{x} + U\mathbf{h}_{i-1} + \mathbf{b})$$

$$\mathbf{y}_i = W' \, \mathbf{h}_i + \mathbf{b}'$$

RNN language model



Mikolov et al. (2010)

Word embeddings

Where do they come from?

Random initialization (when enough training data)

E.g. sample from uniform distribution [-0.01, 0.01]

Supervised pre-training

Train the embeddings first on another task for which you have more data

Unsupervised pre-training

Create your own supervised training instances, e.g. word2vec

Source: Goldberg (2015)

The RNN abstraction

The RNN abstraction

Input:

a sequence of input vectors $\mathbf{x}_{i:j} = \mathbf{x}_i, \ldots, \mathbf{x}_j$

initial state vector $\mathbf{h}_0$

Output:

a sequence of **state vectors** $\mathbf{h}_1, \ldots, \mathbf{h}_n$

$\mathbf{h}_i$ represents the state of the RNN after observing $\mathbf{x}_{1:i}$

Example:

a model for predicting the conditional prob. of an event e given the sequence $\mathbf{x}_{1:i}$

$$p(e = j \mid \mathbf{x}_{1:i}) = \mathrm{softmax}(\mathbf{h}_i W + \mathbf{b})[\,j\,]$$

Source: Goldberg (2015)

The RNN abstraction (2)

We have now defined a **recursive function**:

$RNN(\mathbf{h}_0, \mathbf{x}_{1:n}) = \mathbf{h}_{1:n}$

$\mathbf{h}_i = R(\mathbf{h}_{i-1}, \mathbf{x}_i)$

Here, $\underline{R}$ is a function!

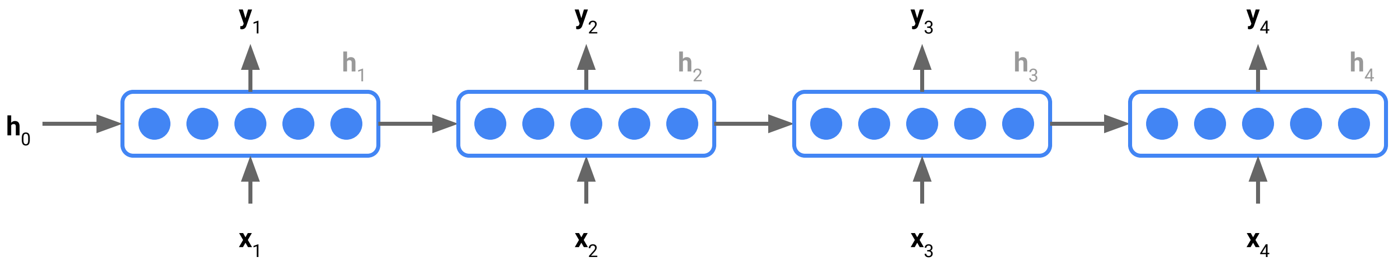
$\mathbf{x}_i \in \mathbb{R}^e$
$\mathbf{h}_i \in \mathbb{R}^d$

$$
\begin{aligned}
\mathbf{h}_4 &= R(\mathbf{h}_3, \mathbf{x}_4) \\
&= R(R(\mathbf{h}_2, \mathbf{x}_3), \mathbf{x}_4) \\
&= R(R(R(\mathbf{h}_1, \mathbf{x}_2), \mathbf{x}_3), \mathbf{x}_4) \\
&= R(R(R(R(\mathbf{h}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3)\ \mathbf{x}_4)
\end{aligned}
$$

During training we hope to set the parameters of R in such a way so that the states $\mathbf{h}_i$ contain useful information for the prediction task.

Source: Goldberg (2015)

The RNN abstraction (3)



Adapted from Goldberg (2015)

Various roles for RNNs

Acceptor

observe final state $\mathbf{h}_n$ and decide on an outcome, e.g. sentiment classification

Encoder

final state $\mathbf{h}_n$ is treated as an *encoding* of the information in the sequence, and is used as additional information together with other signals. e.g. extractive summarization

Transducer

produce an output for each input, e.g. language modeling

Encoder - Decoder

translation! final state $\mathbf{h}_n$ is used as additional input to another RNN

Concrete RNN architectures: Simple RNN

$$RNN(\mathbf{h}_0, \mathbf{x}_{1:n}) = \mathbf{h}_{1:n}$$

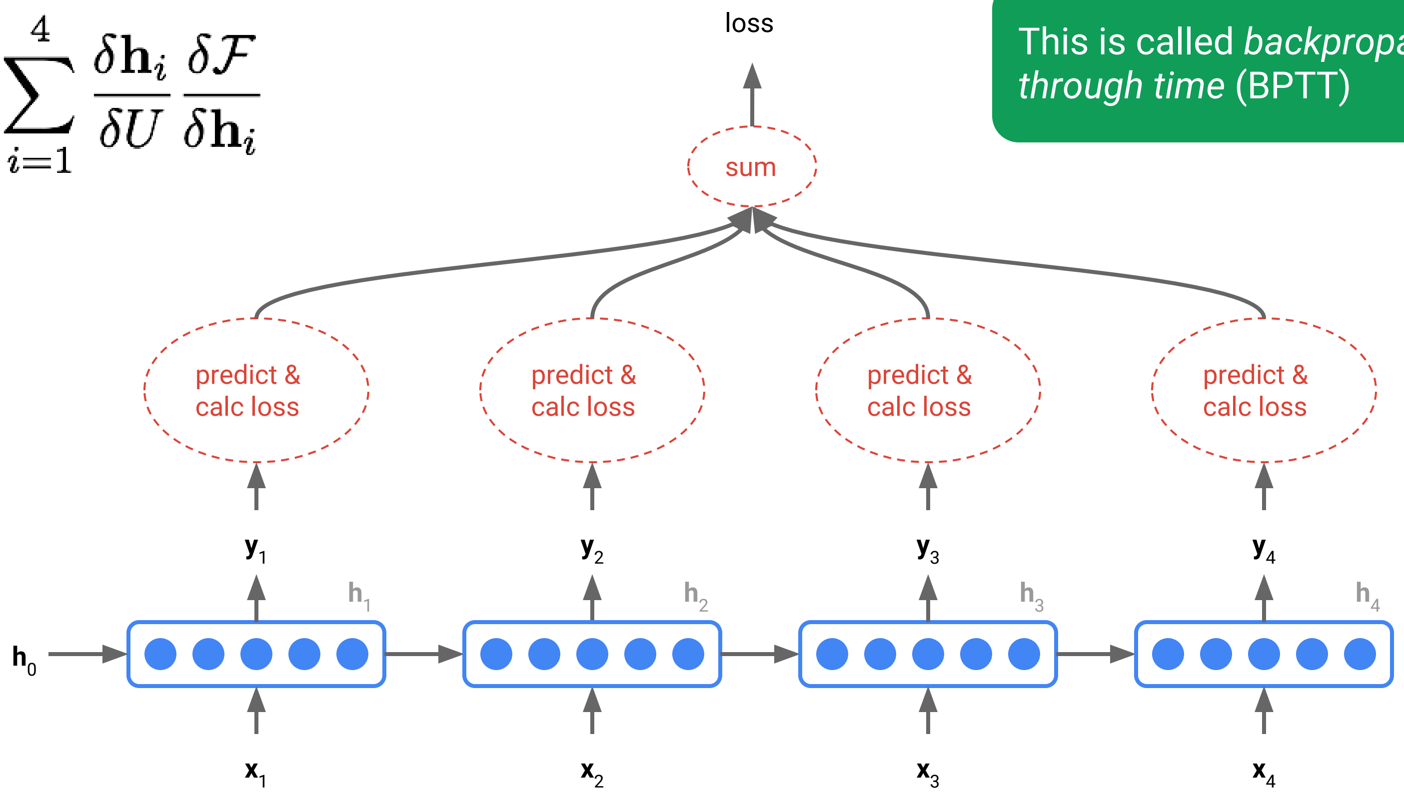
$$\mathbf{h}_i = R(\mathbf{h}_{i-1}, \mathbf{x}_i) = \phi(\mathbf{x}_i W + \mathbf{h}_i U + \mathbf{b})$$

$$\mathbf{x}_i \in \mathbb{R}^e \qquad \mathbf{h}_i \in \mathbb{R}^d \qquad W \in \mathbb{R}^{e \times d} \qquad U \in \mathbb{R}^{d \times d} \qquad \mathbf{b} \in \mathbb{R}^d$$

Training

$$\frac{\delta \mathcal{F}}{\delta U} = \sum_{i=1}^{4} \frac{\delta \mathbf{h}_i}{\delta U} \frac{\delta \mathcal{F}}{\delta \mathbf{h}_i}$$

This is called *backpropagation through time* (BPTT)



Adapted from Goldberg (2015)

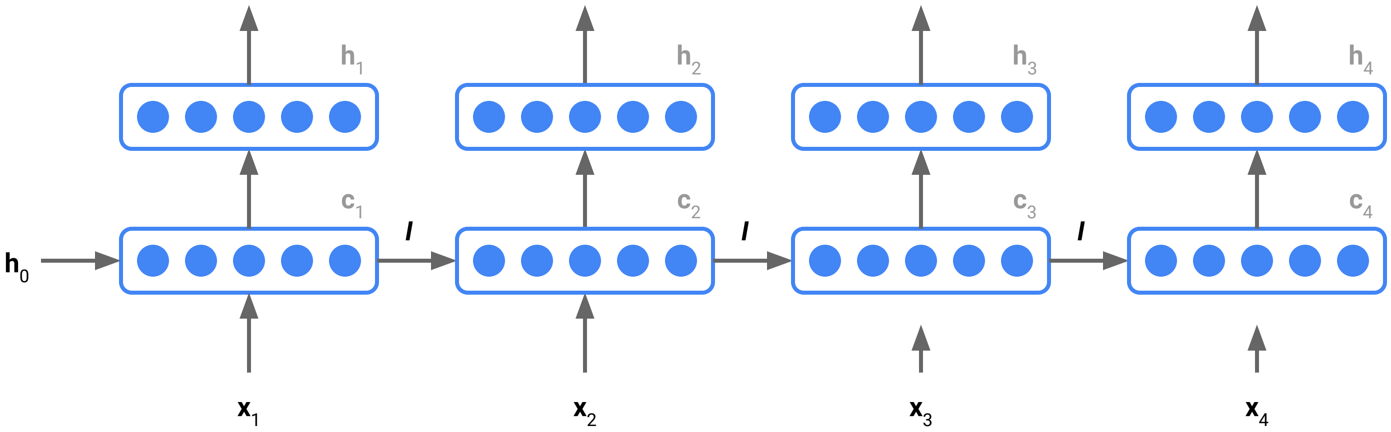
The vanishing & exploding gradient problem

Simple RNNs are hard to train because of the **vanishing gradient** problem.

During backpropagation, error signals (gradients) from later time steps quickly become small, as they repeatedly go through nonlinear functions.

In more rare situations, it is also possible for the gradient to **explode**.

Intuition to solving the vanishing gradient

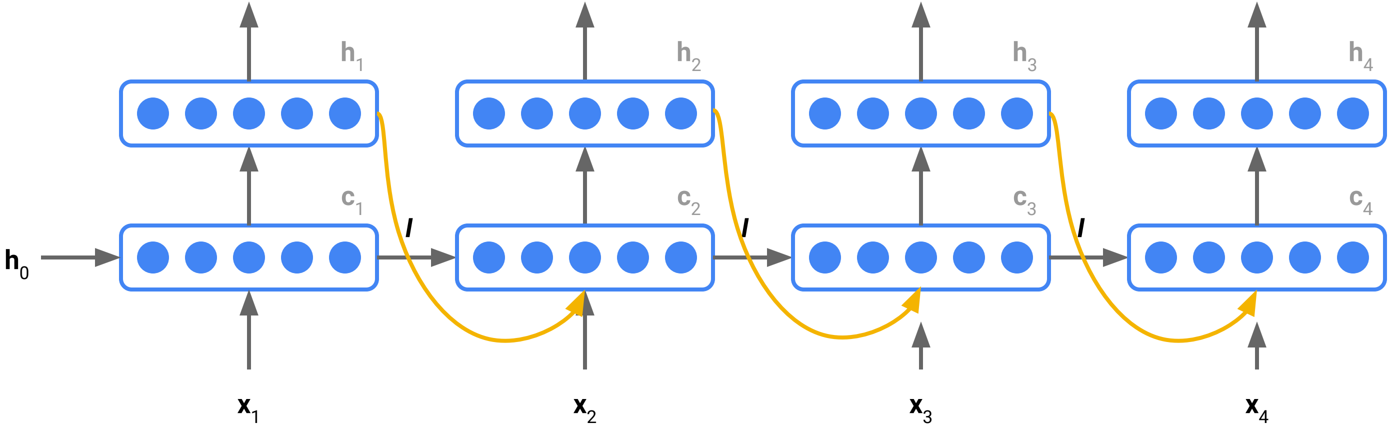


$\mathbf{c}_i = \mathbf{c}_{i-1} + f(\mathbf{x}_i)$ $\quad$ $\mathbf{h}_i = g(\mathbf{c}_i)$ $\quad$ $\frac{\delta \mathbf{c}_i}{\delta \mathbf{c}_{i-1}} = I$

Adapted from Dyer, LxMLS 2016.

Intuition to solving the vanishing gradient (2)

Better gradient propagation is possible when you use **additive** rather than multiplicative/highly non-linear recurrent dynamics



$\mathbf{c}_i = \mathbf{c}_{i-1} + f([\mathbf{x}_i;\mathbf{h}_{i-1}])$ $\quad$ $\mathbf{h}_i = g(\mathbf{c}_i)$ $\quad$ $\frac{\delta \mathbf{c}_i}{\delta \mathbf{c}_{i-1}} = I + \epsilon$

Adapted from Dyer, LxMLS 2016.

Concrete RNN architectures: LSTM

$$LSTM([\mathbf{c}_{i-i}; \mathbf{h}_{i-1}], \mathbf{x}_i) = [\mathbf{c}_i; \mathbf{h}_i]$$

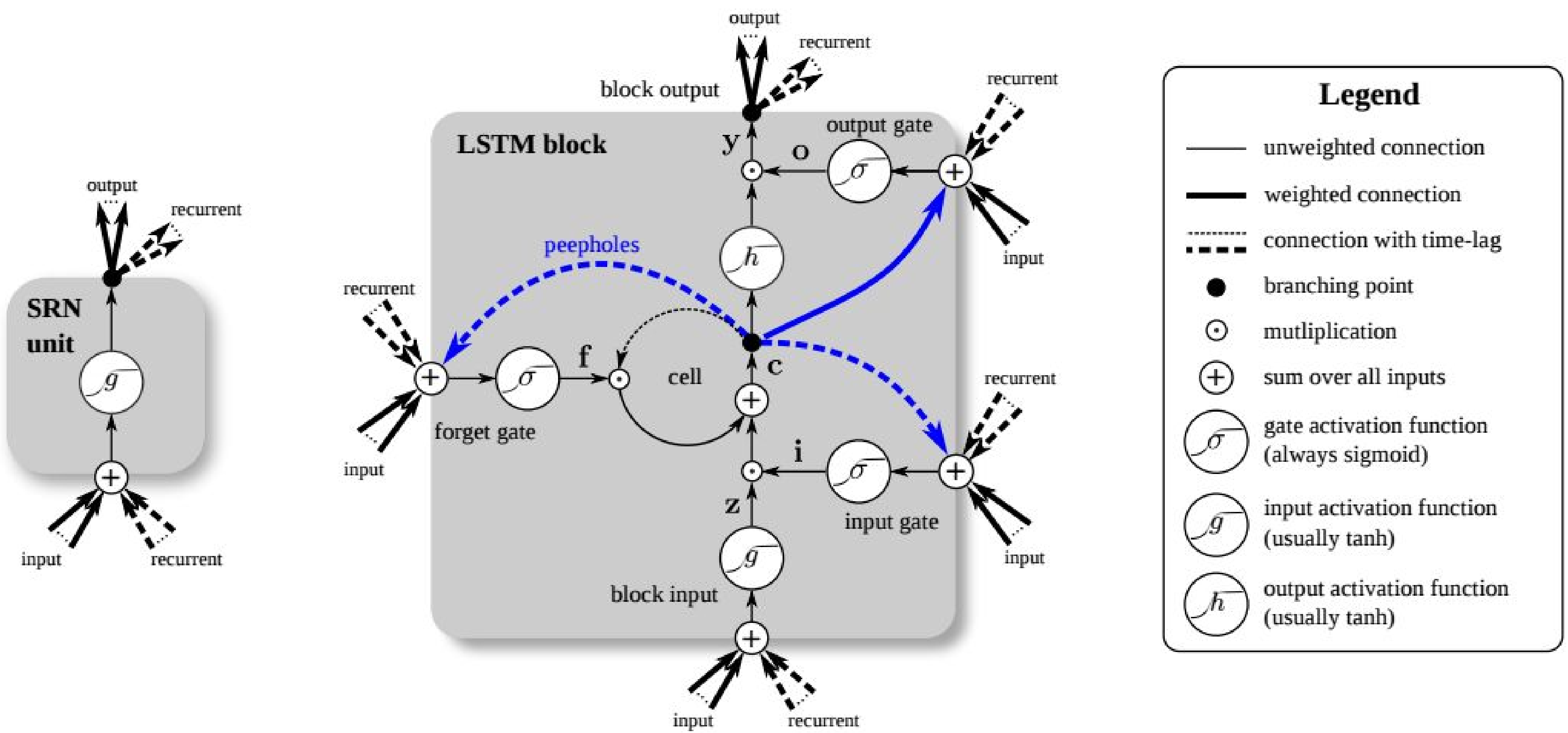
$$\mathbf{c}_i = \mathbf{c}_{i-1} \odot \mathbf{f} + \mathbf{g} \odot \mathbf{i}$$

$$\mathbf{h}_i = \tanh(\mathbf{c}_i) \odot \mathbf{o}$$

$$\mathbf{i} = \sigma(\mathbf{x}_i W^{xi} + \mathbf{h}_{i-1} W^{hi}) \qquad \mathbf{f} = \sigma(\mathbf{x}_i W^{xf} + \mathbf{h}_{i-1} W^{hf}) \qquad \mathbf{o} = \sigma(\mathbf{x}_i W^{xo} + \mathbf{h}_{i-1} W^{ho}) \qquad \mathbf{g} = \tanh(\mathbf{x}_i W^{xg} + \mathbf{h}_{i-1} W^{hg})$$

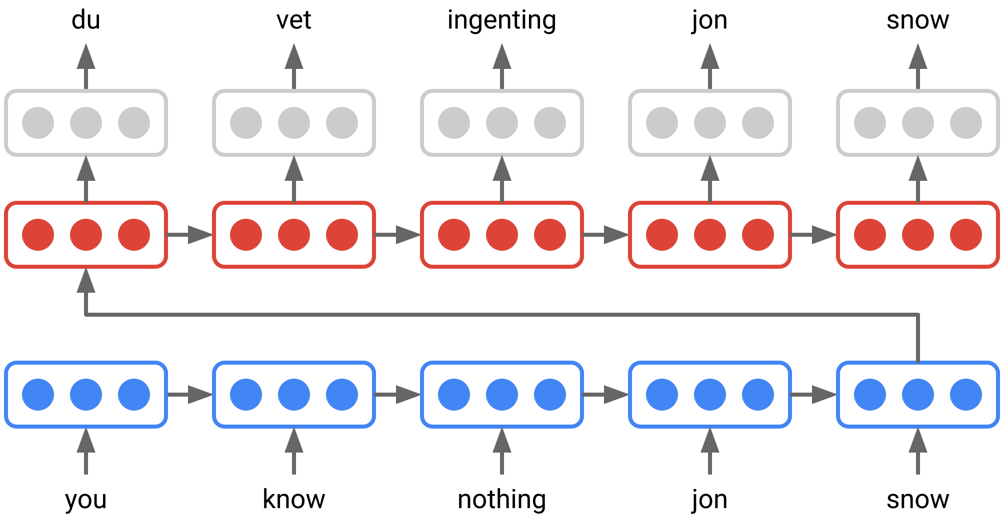
$$\mathbf{x}_i \in \mathbb{R}^e \qquad \mathbf{c}_i, \mathbf{h}_i, \mathbf{i}, \mathbf{f}, \mathbf{o}, \mathbf{g} \in \mathbb{R}^d \qquad W^{x\cdot} \in \mathbb{R}^{e \times d} \quad W^{h\cdot} \in \mathbb{R}^{d \times d}$$

LSTM



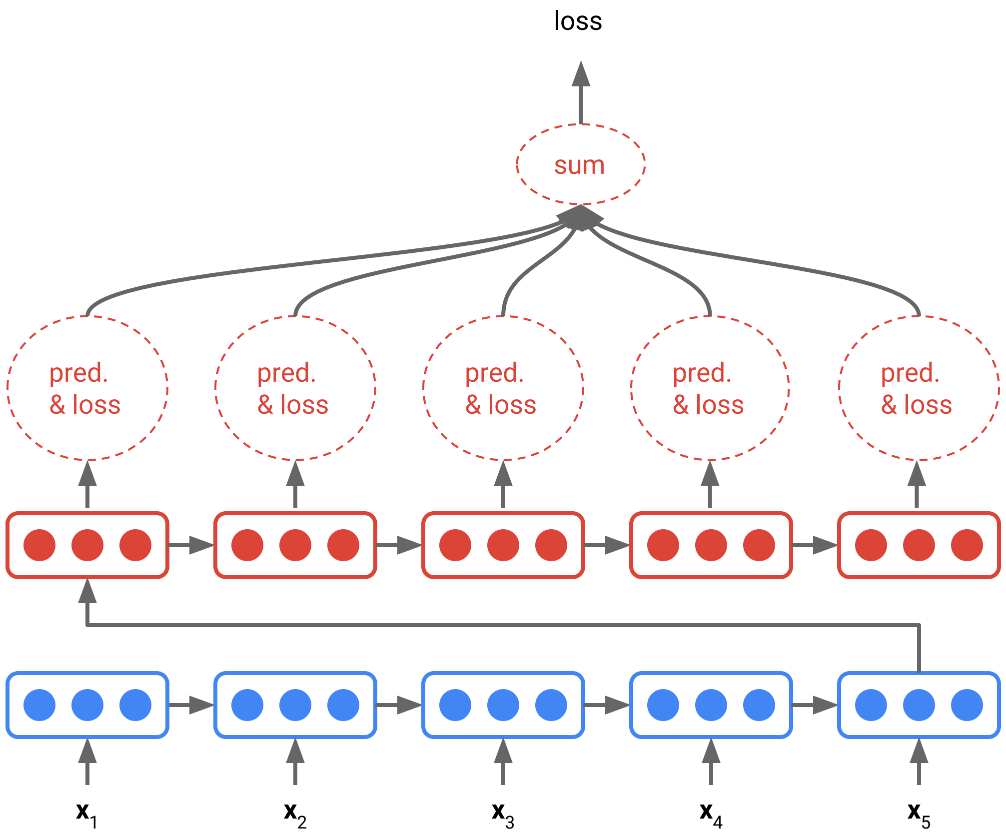
Greff et al. (2015). LSTM: A Search Space Odyssey.

Encoder-Decoder



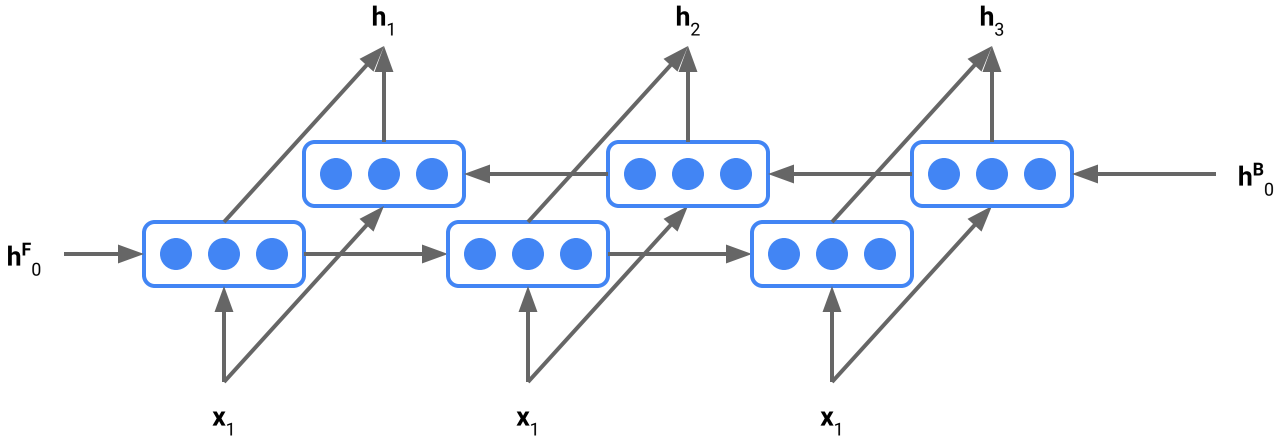
Sutskever et al. (2014), Cho et al. (2014)

Encoder-Decoder Training

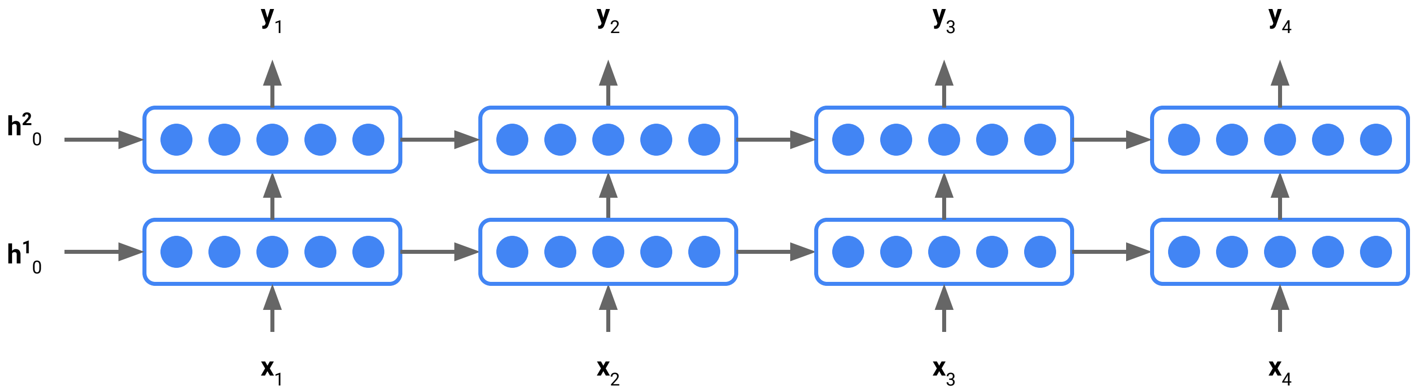


Sutskever et al. (2014), Cho et al. (2014)

Bidirectional RNN

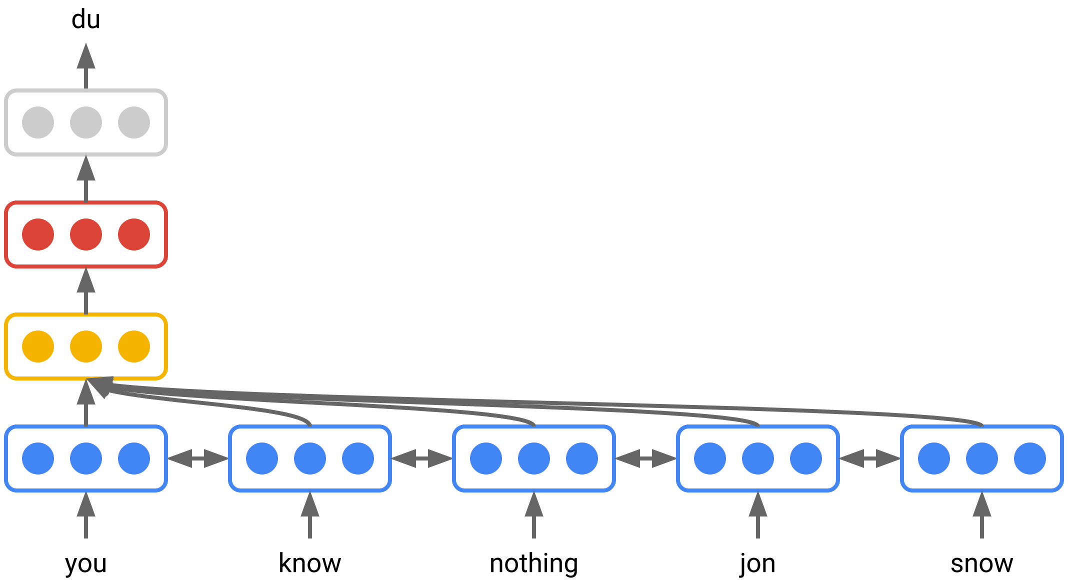


Multi-layer RNN



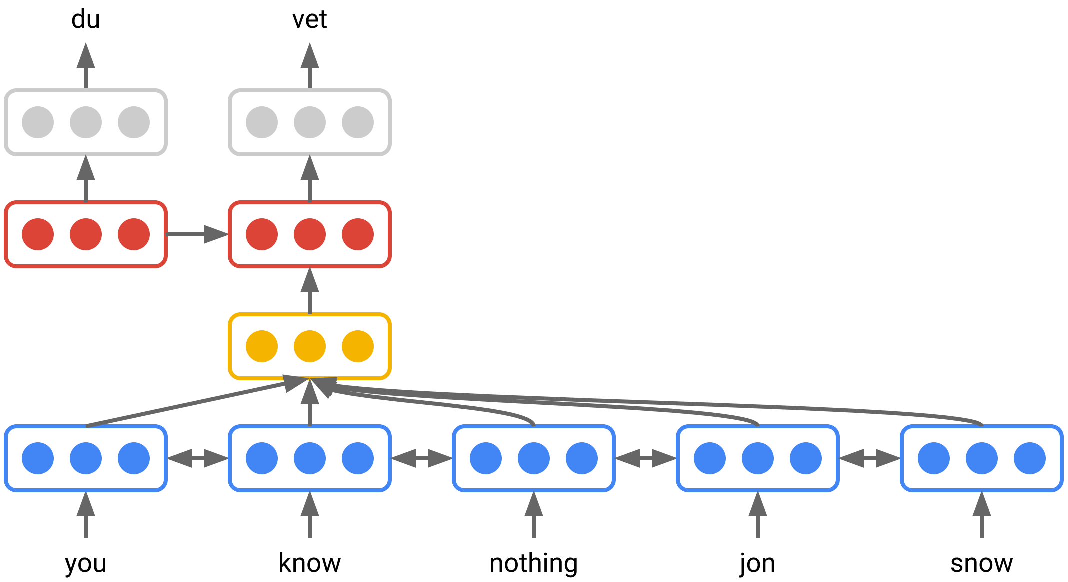
Adapted from Goldberg (2015)

Encoder-decoder with Attention



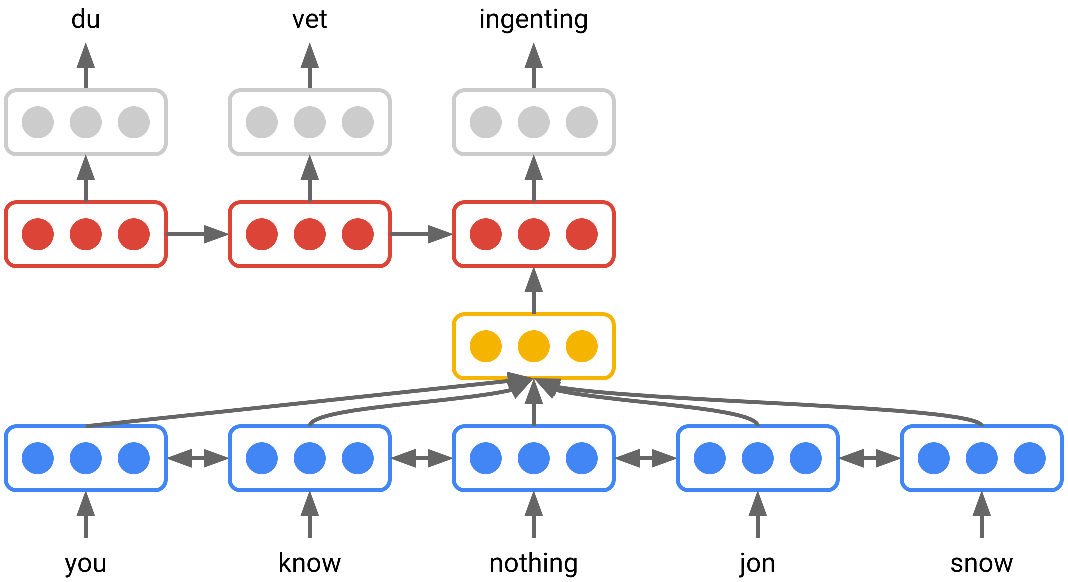
Bahdanau et al. (2015)

Encoder-decoder with Attention

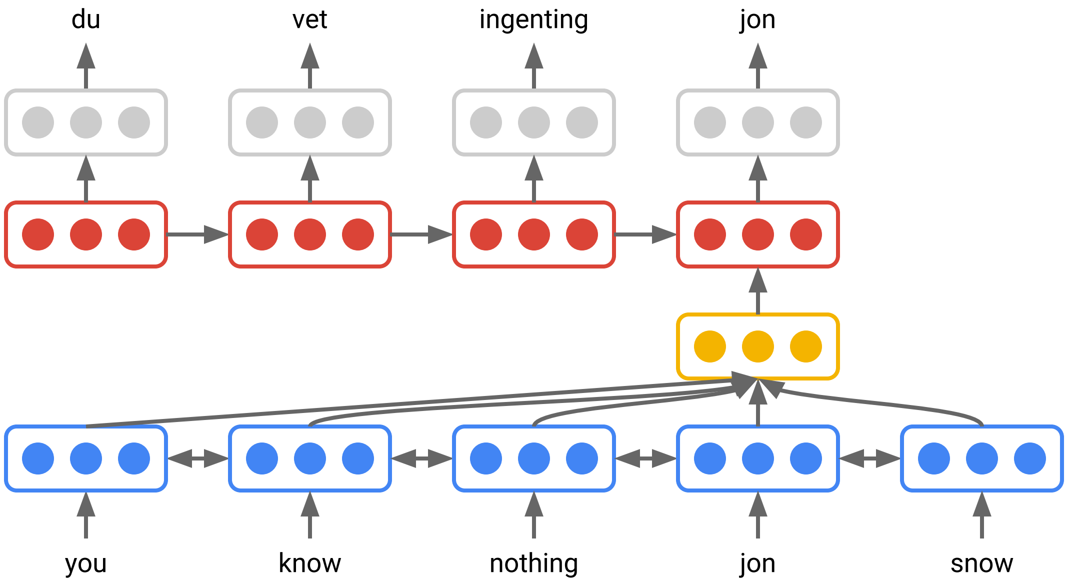


Bahdanau et al. (2015)

Encoder-decoder with Attention

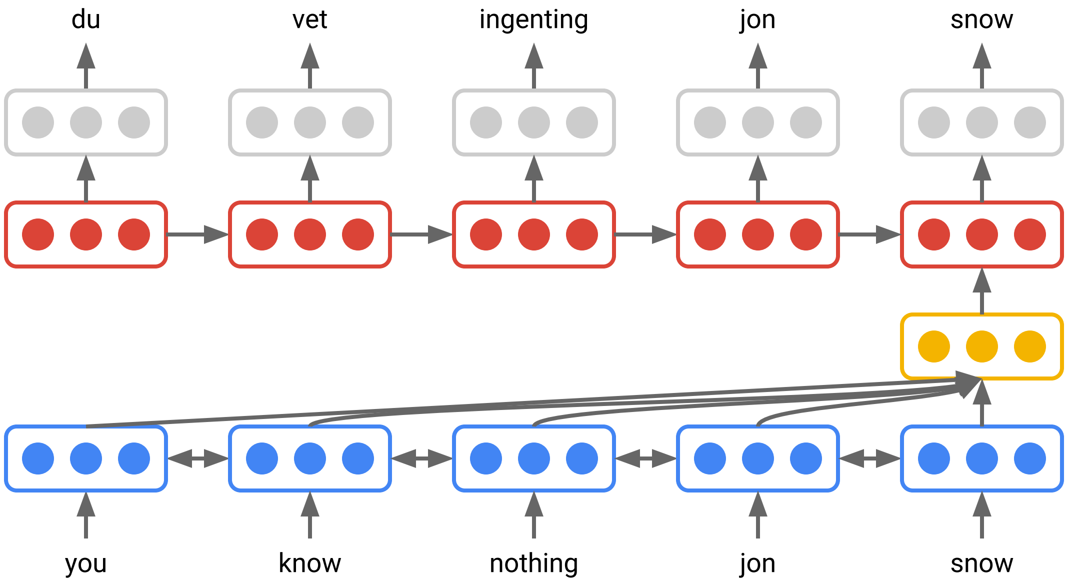


Encoder-decoder with Attention



Bahdanau et al. (2015)

Encoder-decoder with Attention



Bahdanau et al. (2015)

Unknown words

Dealing with unknown words

The *softmax* over the output layer is very expensive!

In practice we need to use a **limited vocabulary**, e.g. the top 50 000 words.

Infrequent words are now translated as **“UNK”**

Not so ideal! What can we do about this?

One solution: Byte Pair Encoding

Start with a vocabulary of **characters**

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, w, s, t, i, d

Adapted from Sennrich

One solution: Byte Pair Encoding

Start with a vocabulary of **characters**

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 **es** t </w>
3 w i d **es** t </w>

Vocabulary

l, o, w, e, r, n, w, s, t, i, d,
es

Add pair (e, s) with frequency 9

One solution: Byte Pair Encoding

Start with a vocabulary of **characters**

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 **est** </w>
3 w i d **est** </w>

Vocabulary

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

Add pair (es, t) with frequency 9

One solution: Byte Pair Encoding

Start with a vocabulary of **characters**

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

Dictionary

5 lo w </w>
2 lo w e r </w>
6 n e w est </w>
3 w i d est </w>

Vocabulary

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

Add pair (l, o) with frequency 7

Adapted from Sennrich (2016) / ACL NMT

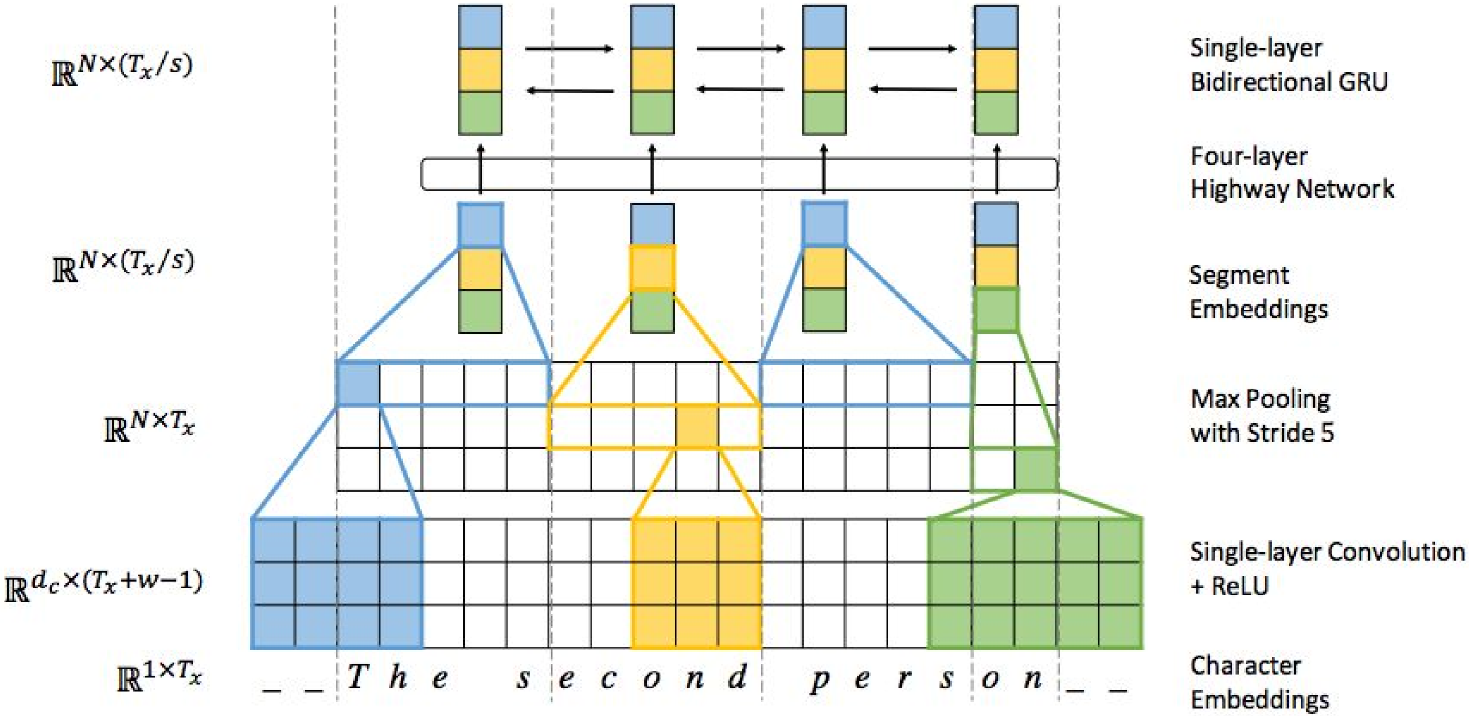
Example: WMT17 English-Latvian

source:
critics said the government funding described by the Los Angeles-based ...

target:
kritiķi apgalvo , ka
Losandželosas
metropoles
ūdensapgādes pārvaldes ...

target_bpe:
krit@@ iķ@@ i apgalv@@ o , ka
L@@ os@@ and@@ ž@@ el@@ os@@ as
me@@ tr@@ op@@ ol@@ es
ūden@@ sa@@ p@@ gād@@ es pārvaldes ...

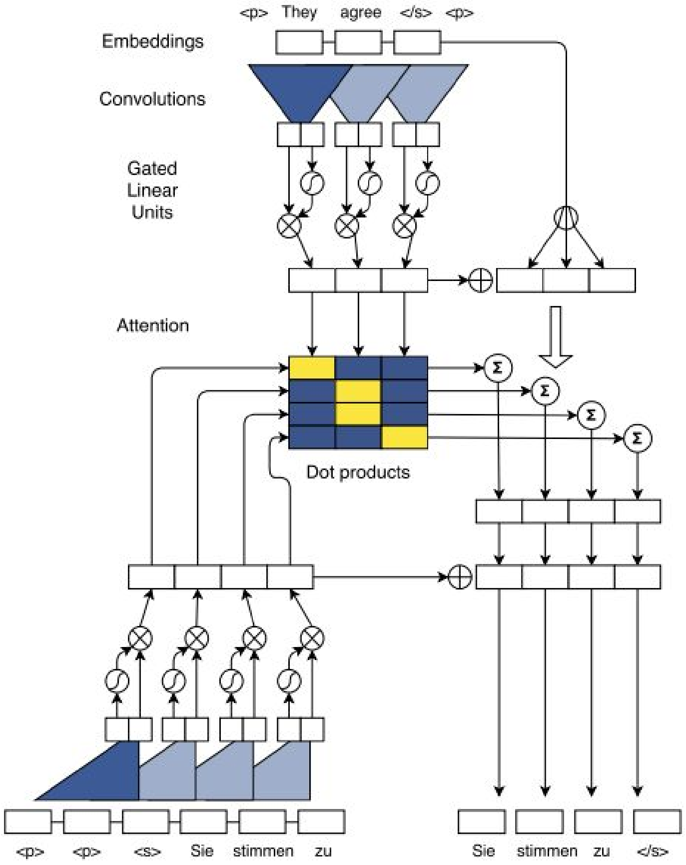
Another solution: Character-based NMT



Source: Lee et al. (2016). Fully Character-Level Neural Machine Translation without Explicit Segmentation.

Do we need to use RNNs?

Convolutions instead of RNNs



References

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." NIPS, 2014.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." ICLR, 2015. arXiv:1409.0473.