Deep Learning

Episode 5

Recurrent Neural Networks II

Yandex Data Factory

LAMBDA

British Hedgehog Preservation Society

Homework assignment TBA
{we fucked up}

Last wave of homeworks (last 5 days) to be checked until the end of the weekend


Ура,
каникулы!

Deep Learning

Episode 5

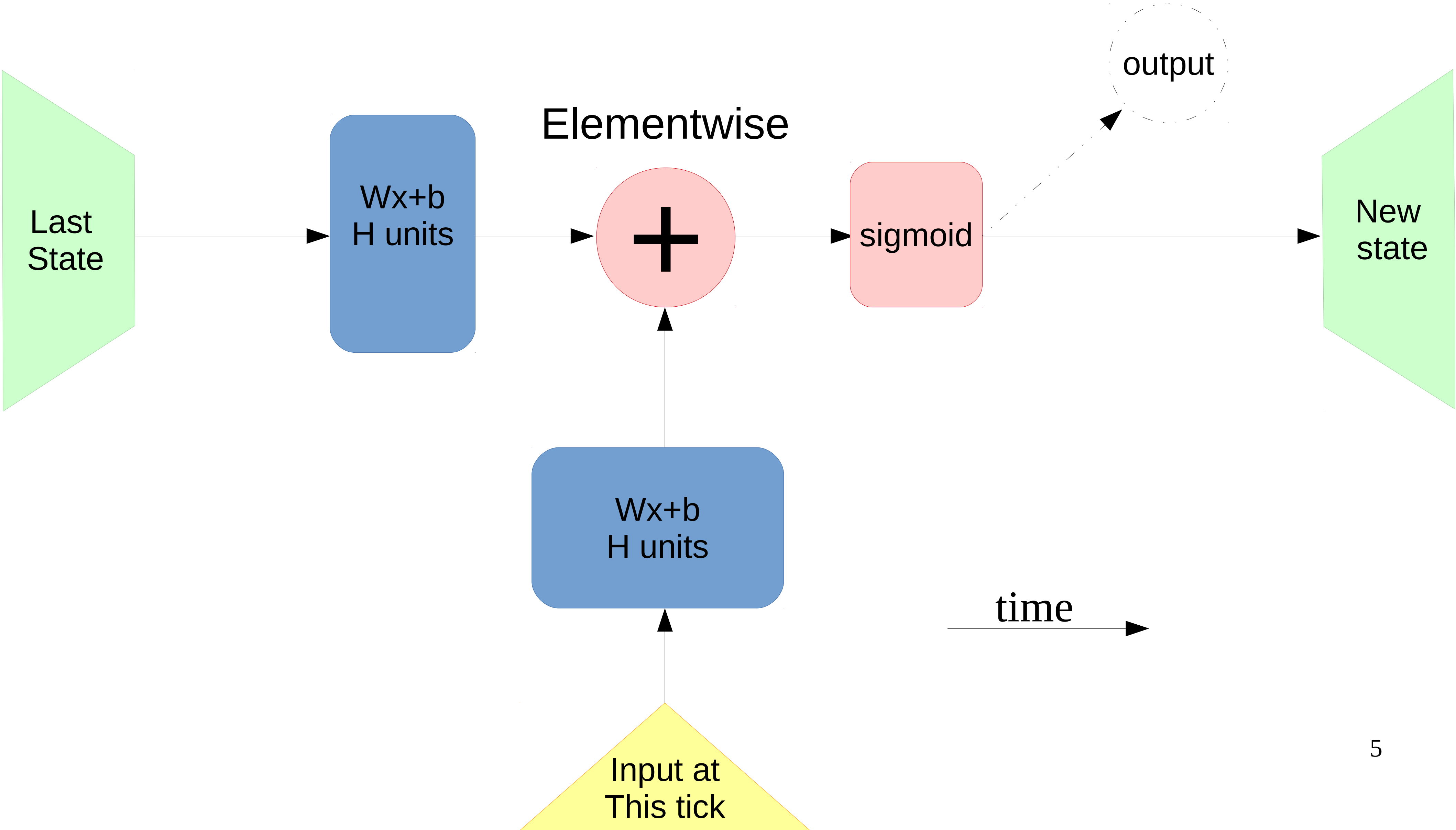
Recurrent Neural Networks II

Yandex Data Factory

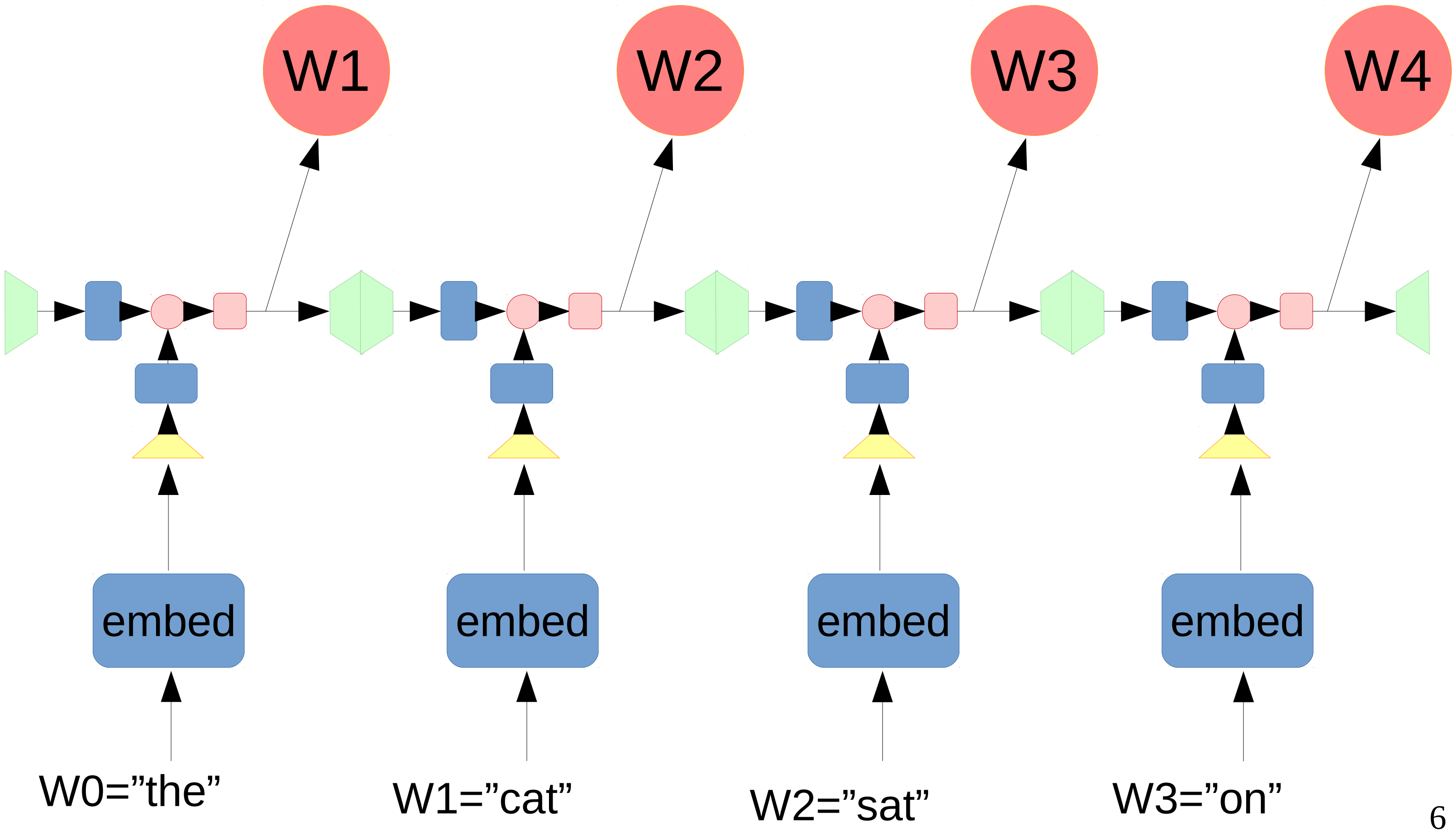
LAMBDA

British Hedgehog Preservation Society

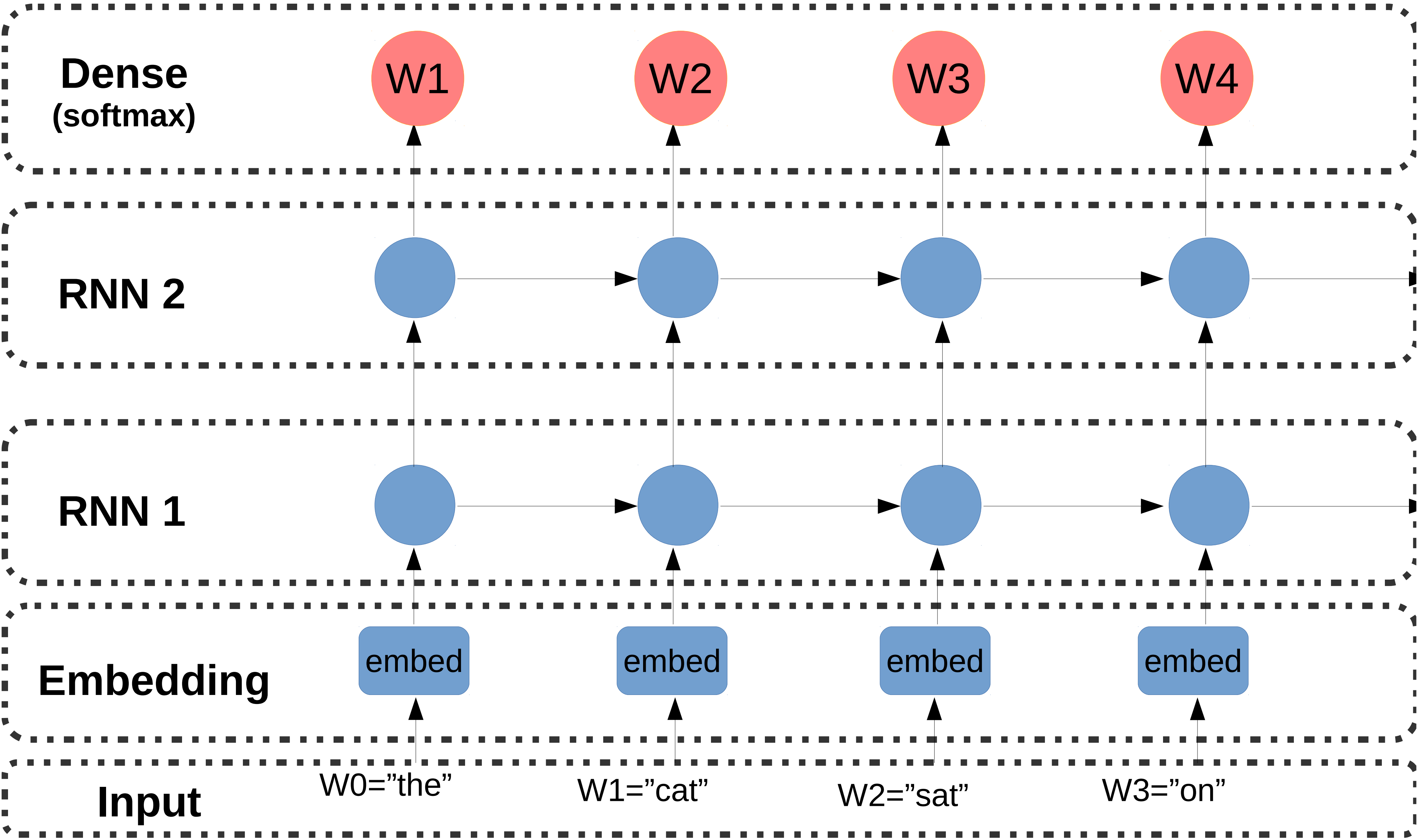
Recap: rnn step



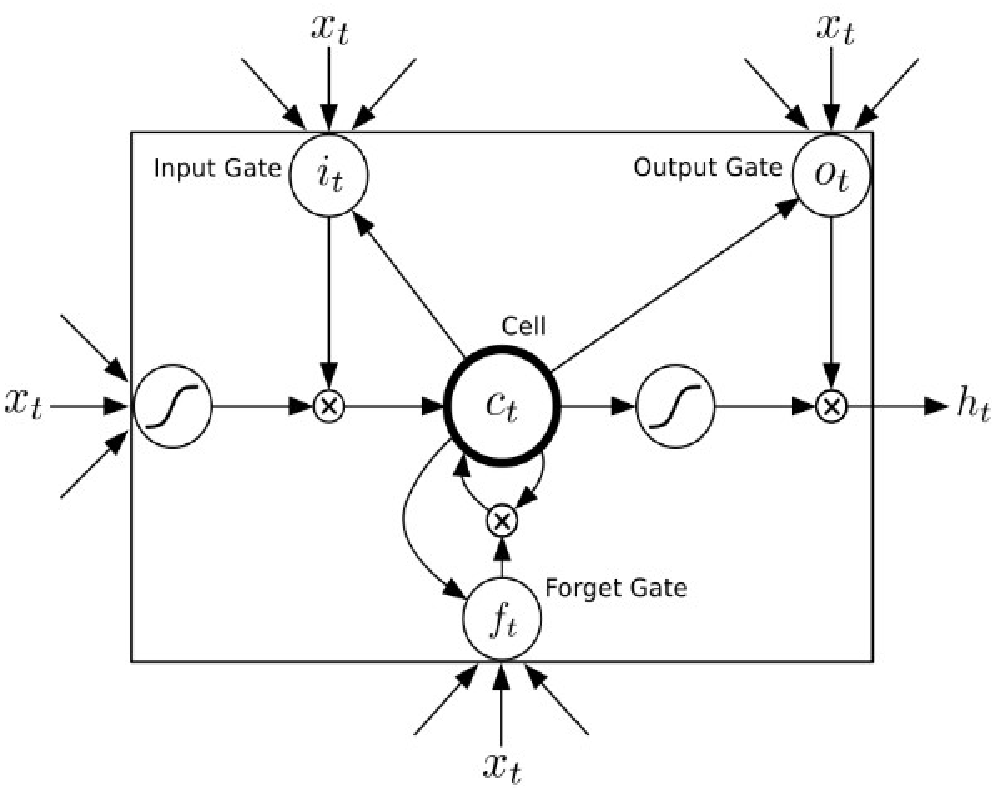
Recap: rnn layer



Recap: multi-layer models



Recap: ~~OMFG~~ LSTM

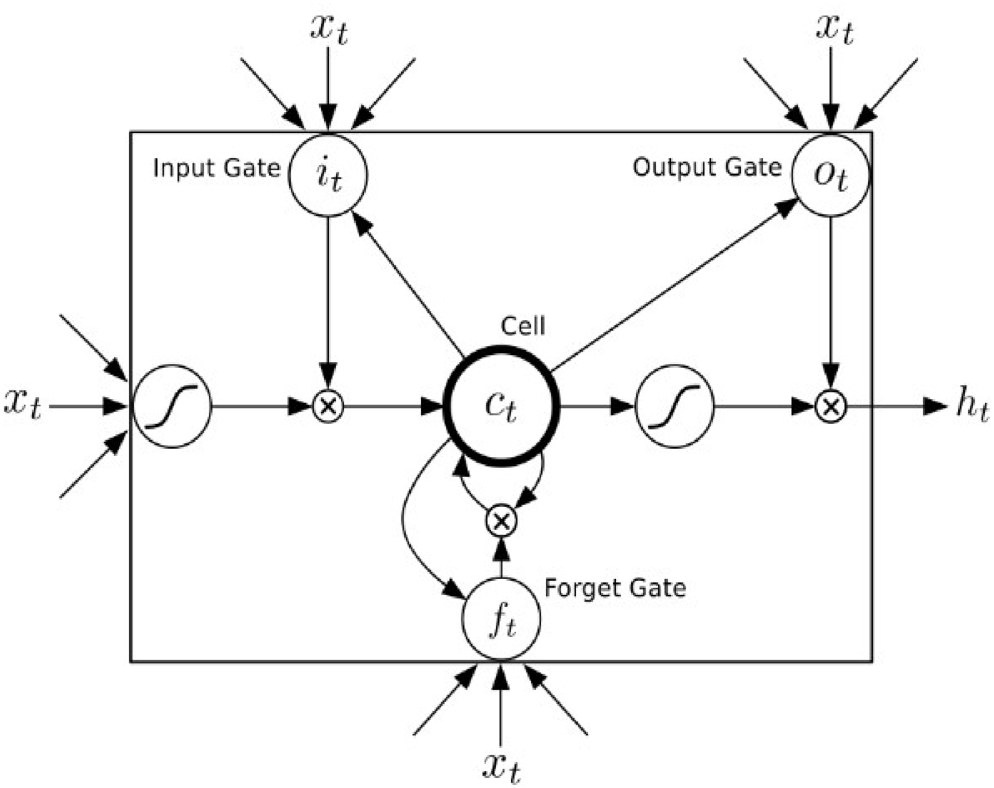


Problem to fix:
- Vanishing gradients

2 hidden states:
- Cell (“private” state)
- Output (“public” state)

4 blocks:
- Update
- Forget gate
- Input gate
- Output gate

Recap: LSTM



$$i_t = Sigm(\theta_{xi} x_t + \theta_{hi} h_{t-1} + b_i)$$

$$f_t = Sigm(\theta_{xf} x_t + \theta_{hf} h_{t-1} + b_f)$$

$$o_t = Sigm(\theta_{xo} x_t + \theta_{ho} h_{t-1} + b_o)$$

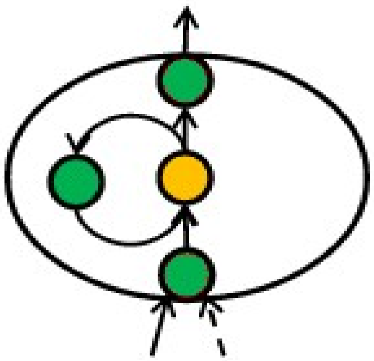
$$g_t = Tanh(\theta_{xg} x_t + \theta_{hg} h_{t-1} + b_g)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t$$

$$h_t = o_t \otimes Tanh(c_t)$$

LSTM: not a monster

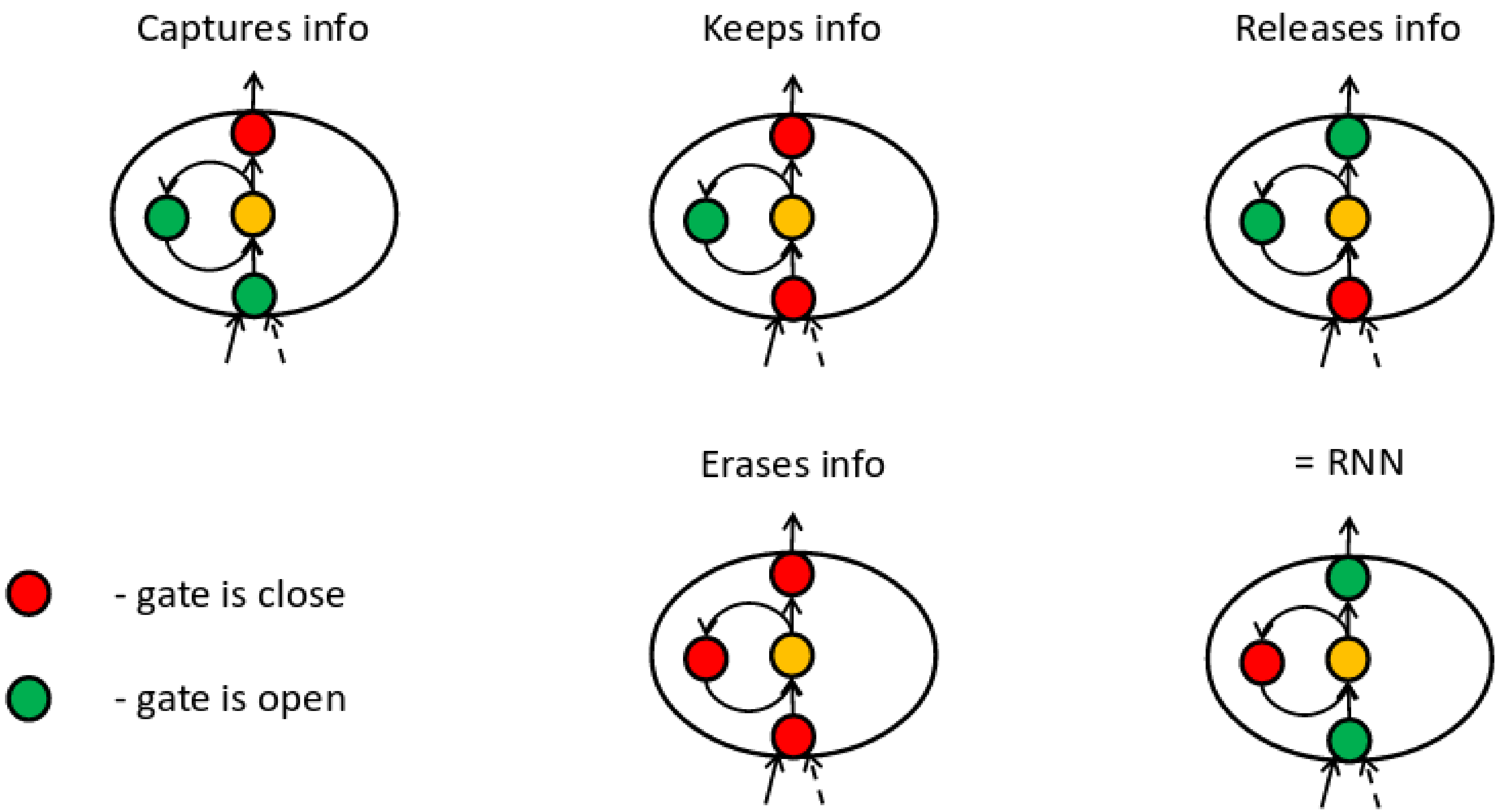
LSTM cell:



$$i_t = Sigm(\theta_{xi} x_t + \theta_{hi} h_{t-1} + b_i)$$
$$f_t = Sigm(\theta_{xf} x_t + \theta_{hf} h_{t-1} + b_f)$$
$$o_t = Sigm(\theta_{xo} x_t + \theta_{ho} h_{t-1} + b_o)$$
$$g_t = Tanh(\theta_{xg} x_t + \theta_{hg} h_{t-1} + b_g)$$
$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t$$
$$h_t = o_t \otimes Tanh(c_t)$$

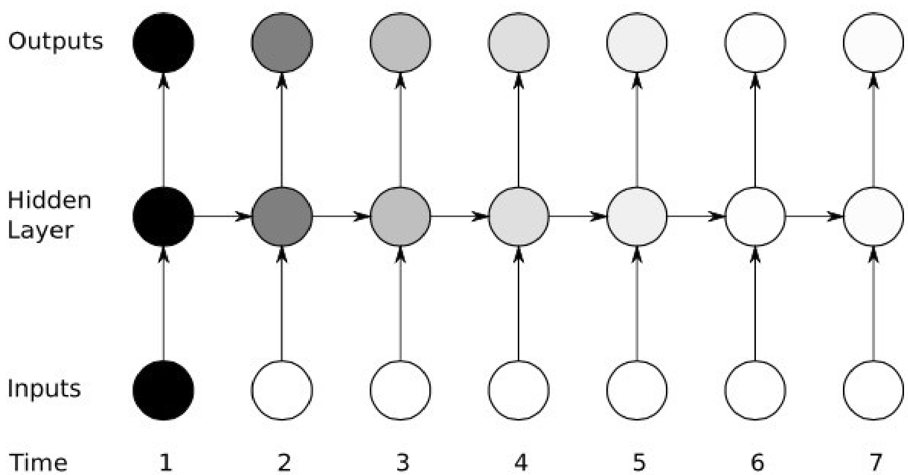
[Pictures: E Lobacheva, D Vetrov]

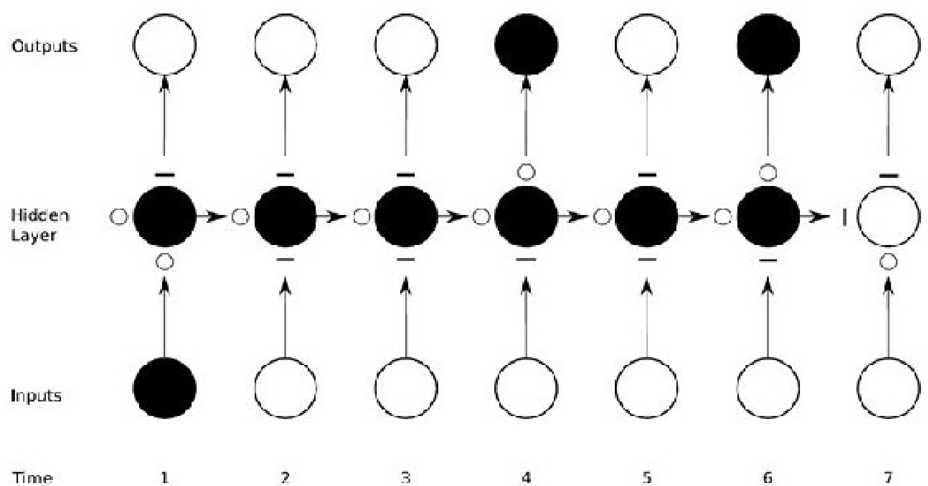
LSTM: not a monster



[Pictures: E Lobacheva, D Vetrov]

LSTM vs RNN





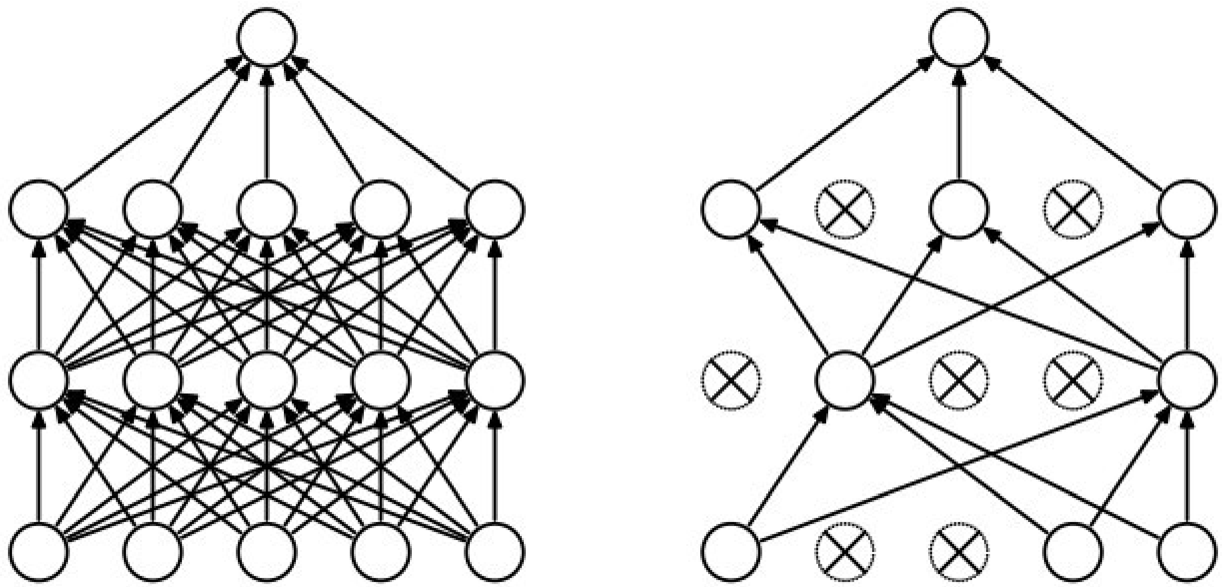
RNN activation

LSTM activation

Recurrent dropout

Wait, what does regular dropout do in the first place?

Recap: dropout

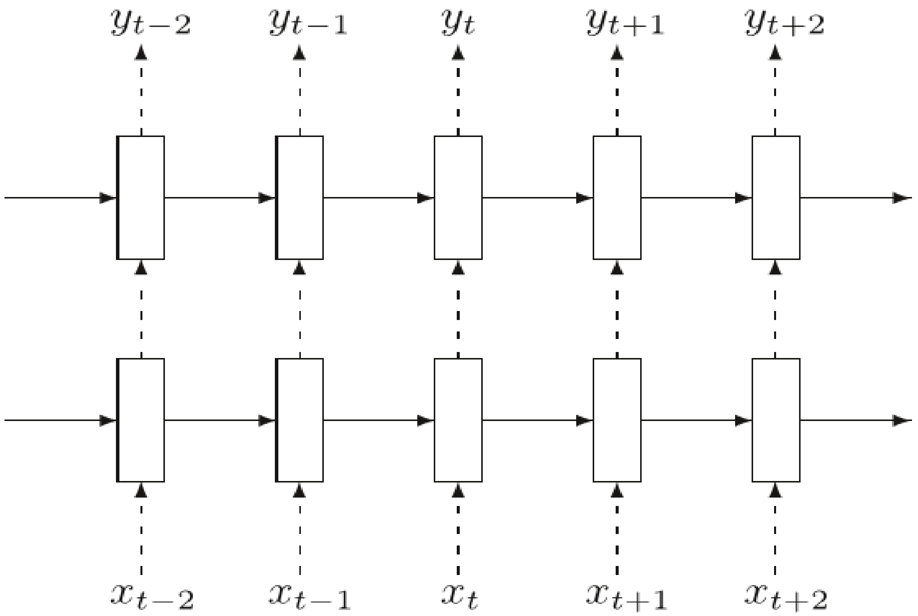


(a) Standard Neural Net

(b) After applying dropout.

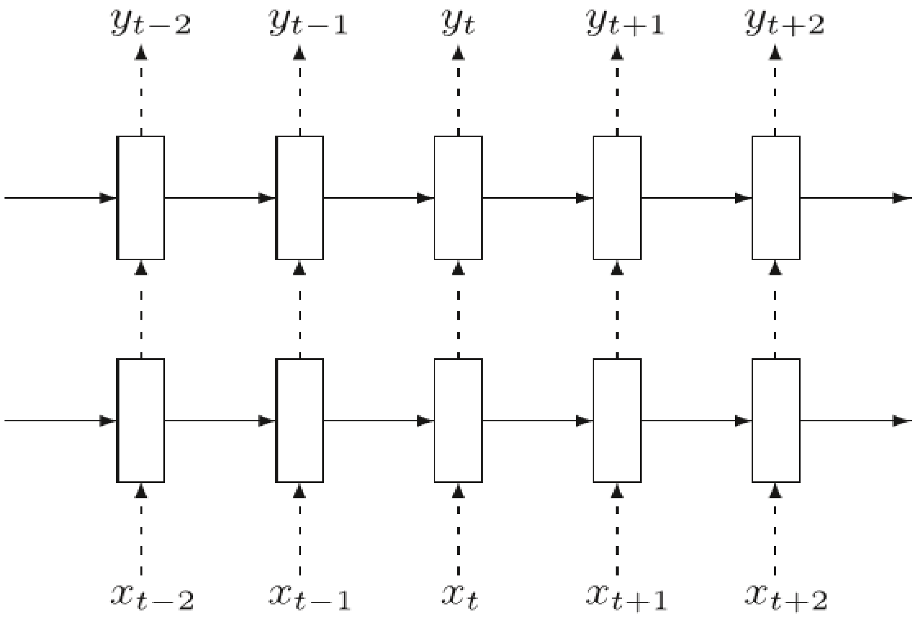
- A way to prevent overfitting
- Randomly turn off some fraction of neurons on each training iteration

Recurrent dropout



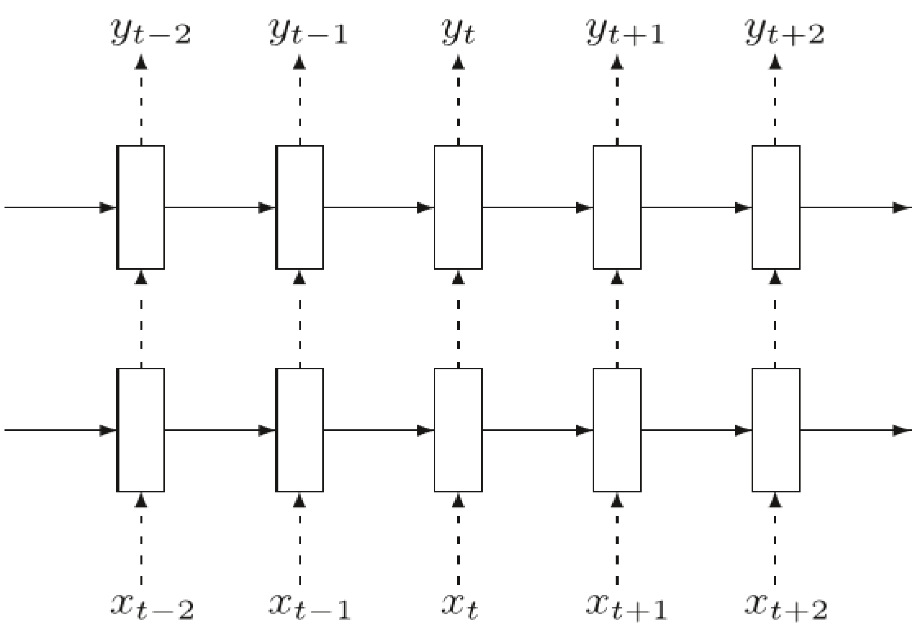
- Say, we dropout 10% activations on each tick
- The dropouts are different at each tick.

Recurrent dropout



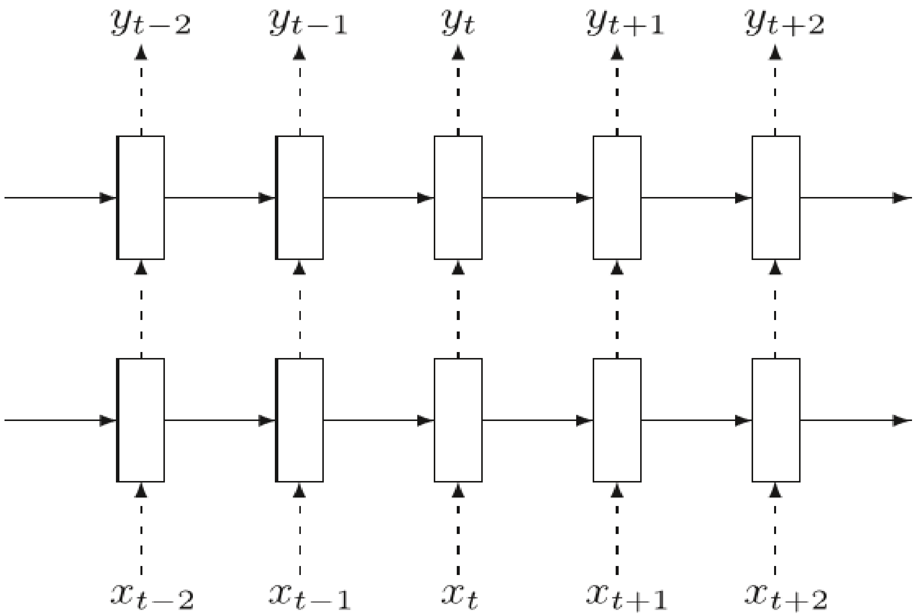
- Say, we dropout 10% activations on each tick
- The dropouts are different at each tick.
- **Trivia:** What is the probability that a one cell will NOT be dropped out:
  - over 10 ticks?
  - over 100 ticks?

Recurrent dropout



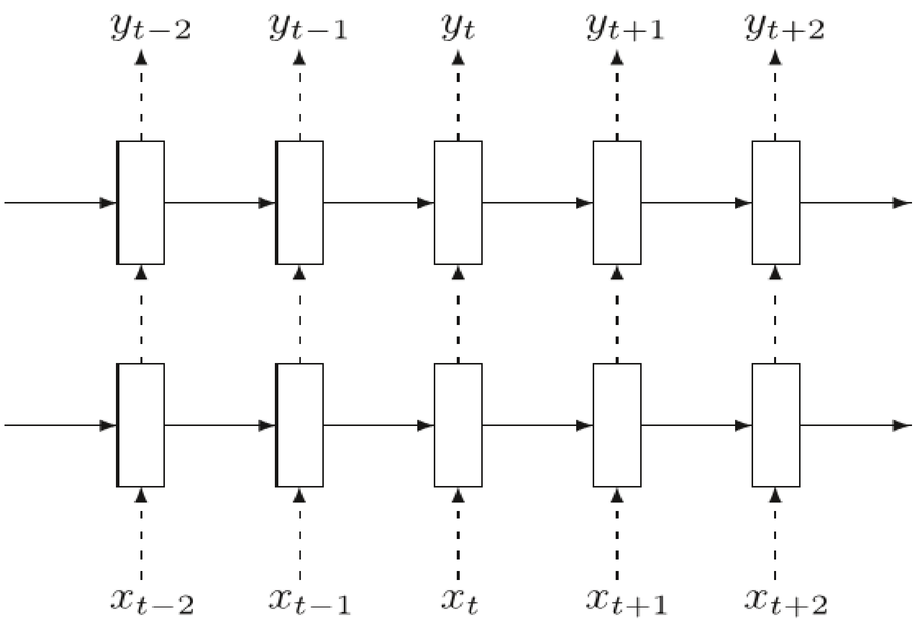
- Say, we dropout 10% activations on each tick
- The dropouts are different at each tick.
- **Trivia:** What is the probability that a one cell will NOT be dropped out:
  - over 10 ticks? **~0.35**
  - over 100 ticks? **~10^-5**
  - **How** does this influence long-term learning?

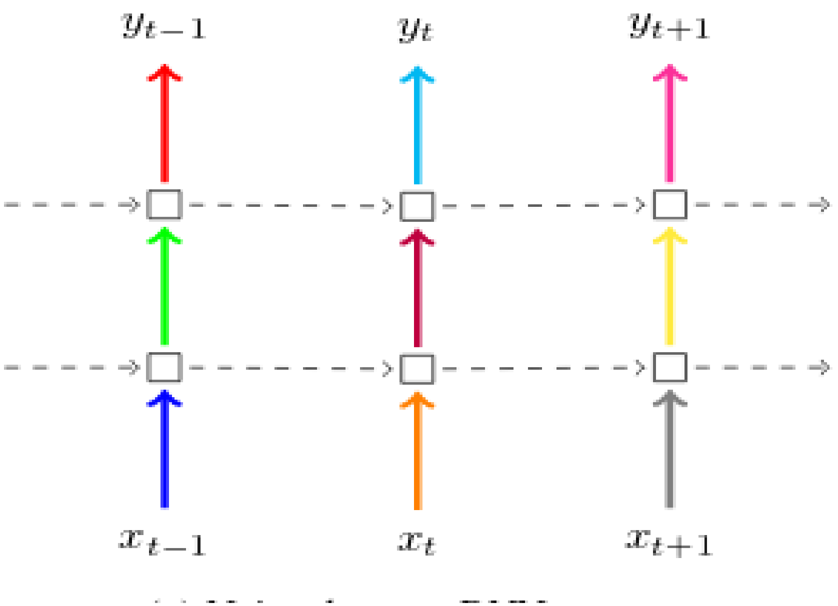
Recurrent dropout: Naive



- **Naive dropout** inside recurrence **sucks**.
- **Idea 1:** Let us only use dropout on non-recurrent connections (vertical arrows)

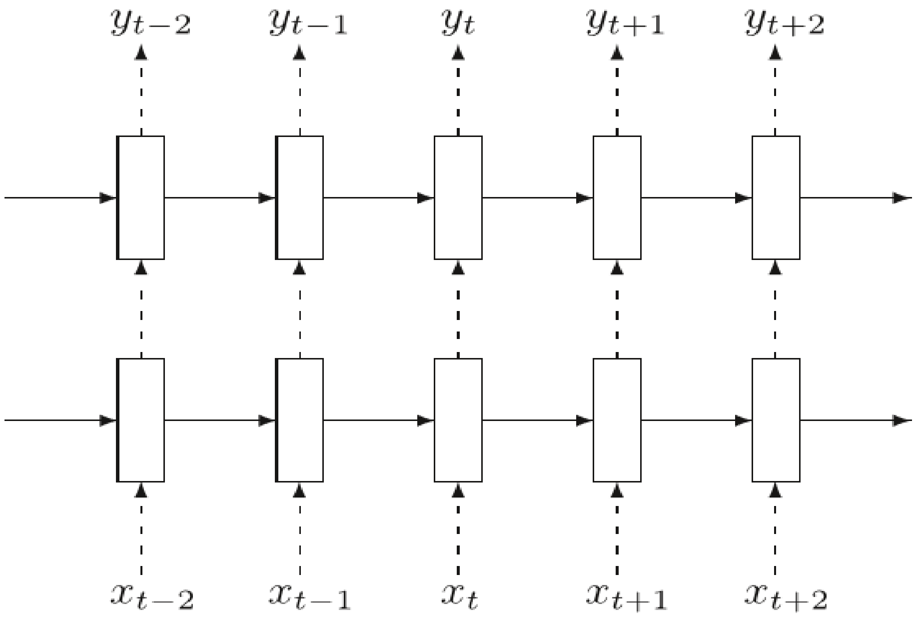
Recurrent dropout: Naive





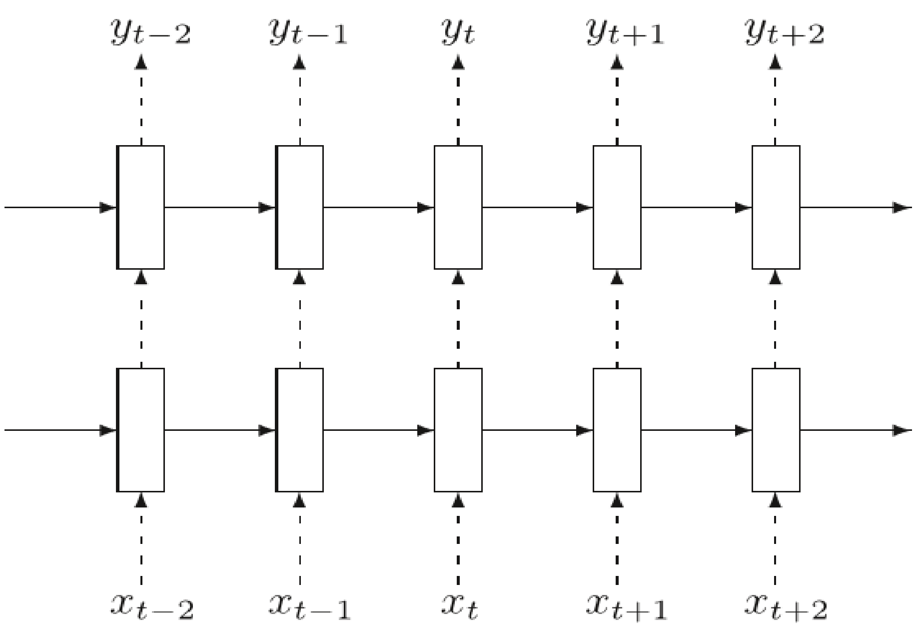
- Random dropout mask (each one is different

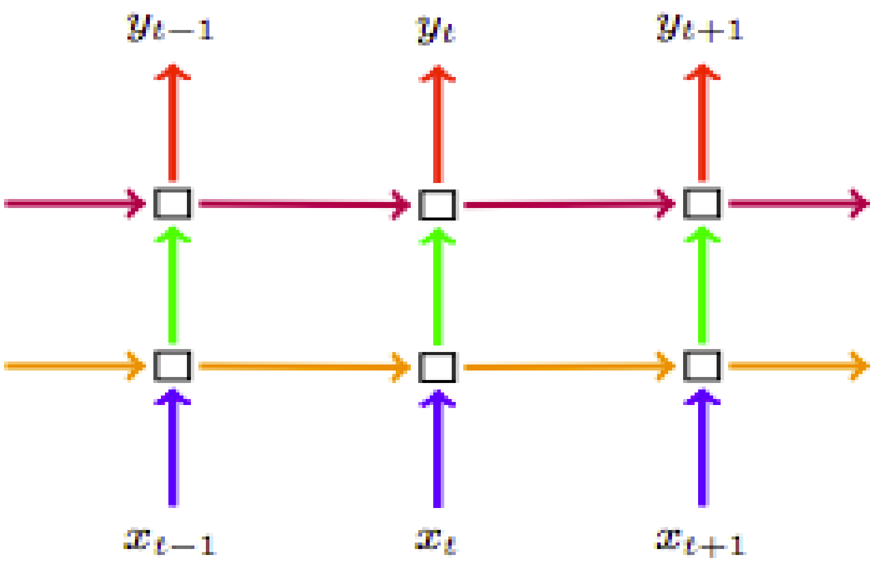
Recurrent dropout: Variational



- Naive dropout inside recurrence sucks.
- **Idea 2:** Let us share the same dropout mask between recurrent connections.
  - Neurons stay alive all through the sequence.
- https://arxiv.org/abs/1512.05287

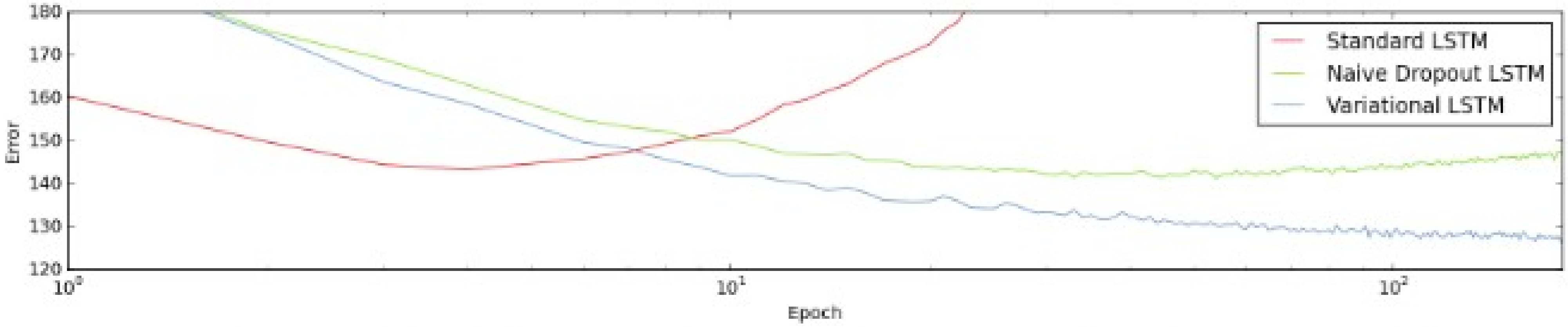
Recurrent dropout: Variational





- **Arrows:** Each color denotes some dropout mask
 (same color = same dropout)
- https://arxiv.org/abs/1512.05287

Recurrent dropout pic-cha



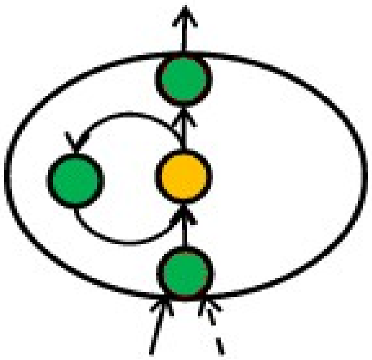
Task: language modelling
(Penn Treebank dataset)

- Naive = only dropout non-recurrent
- Variational = use same masks

- https://arxiv.org/abs/1512.05287

Recurrent Batch Norm

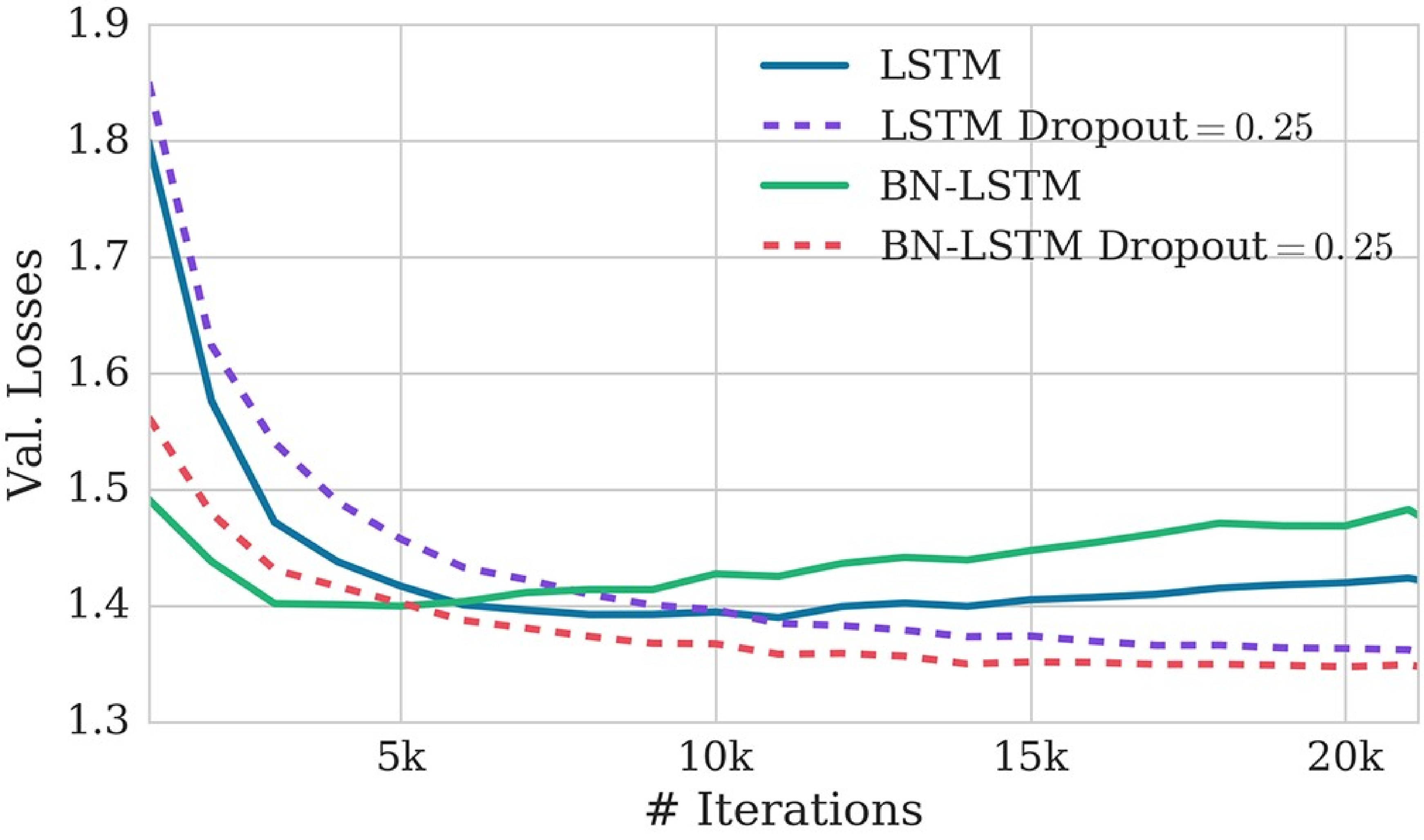
LSTM cell:



$$
\begin{aligned}
\begin{pmatrix} \tilde{\mathbf{f}}_t \\ \tilde{\mathbf{i}}_t \\ \tilde{\mathbf{o}}_t \\ \tilde{\mathbf{g}}_t \end{pmatrix} &= \mathrm{BN}(\mathbf{W}_h \mathbf{h}_{t-1}; \gamma_h, \beta_h) + \mathrm{BN}(\mathbf{W}_x \mathbf{x}_t; \gamma_x, \beta_x) + \mathbf{b} \\
\mathbf{c}_t &= \sigma(\tilde{\mathbf{f}}_t) \odot \mathbf{c}_{t-1} + \sigma(\tilde{\mathbf{i}}_t) \odot \tanh(\tilde{\mathbf{g}}_t) \\
\mathbf{h}_t &= \sigma(\tilde{\mathbf{o}}_t) \odot \tanh(\mathrm{BN}(\mathbf{c}_t; \gamma_c, \beta_c))
\end{aligned}
$$

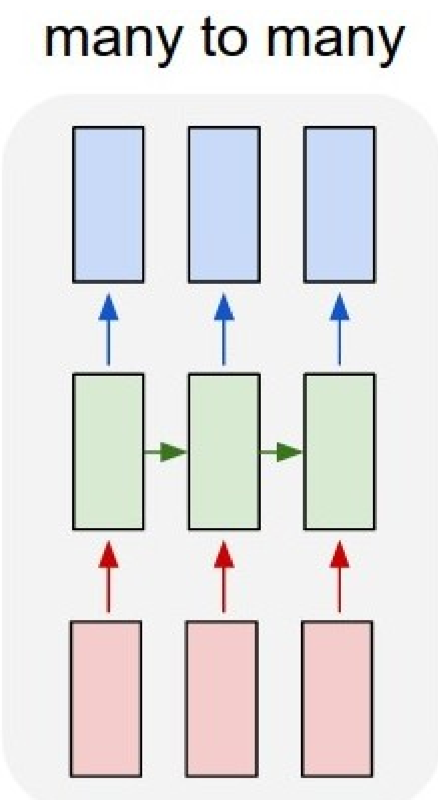
https://arxiv.org/abs/1603.09025

Recurrent Batch Norm



https://arxiv.org/abs/1603.09025

Recurrent Architectures: regular

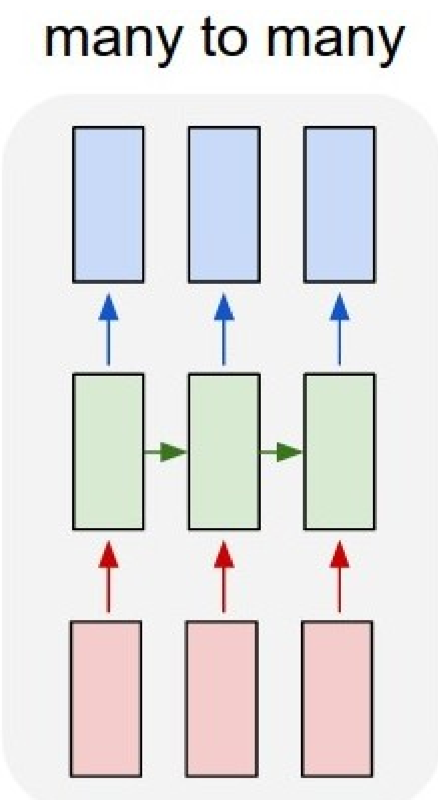


- Read sequence
- Predict sequence of answers at each tick

Tasks:
- Language model
- POS Tagging

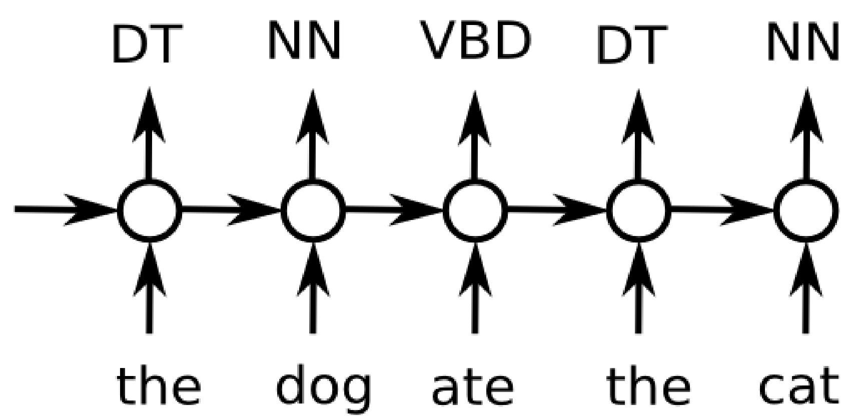
How to implement?
- See last week

Recurrent Architectures: regular



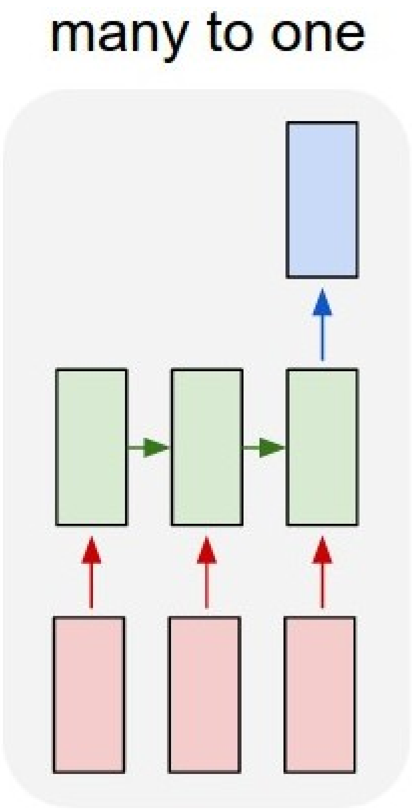
- Read sequence
- Predict sequence of answers at each tick

POS tagging



Why RNN?

Recurrent Architectures: Encoder

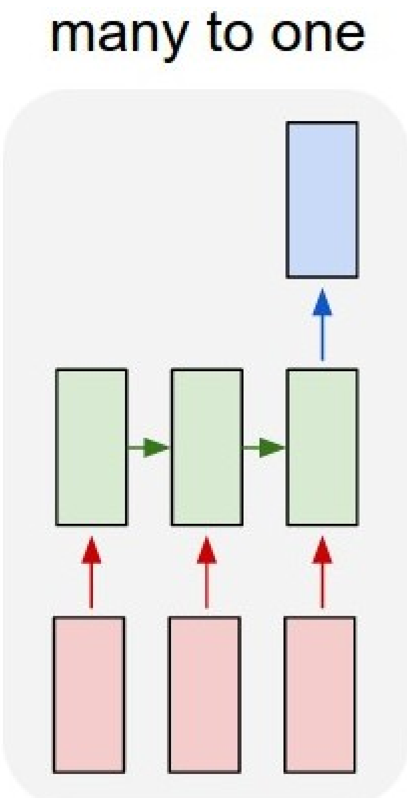


Encoder
- Read sequence
- Predict once

Tasks:
- **?!**

How to implement?
- **?!**

Recurrent Architectures: Encoder

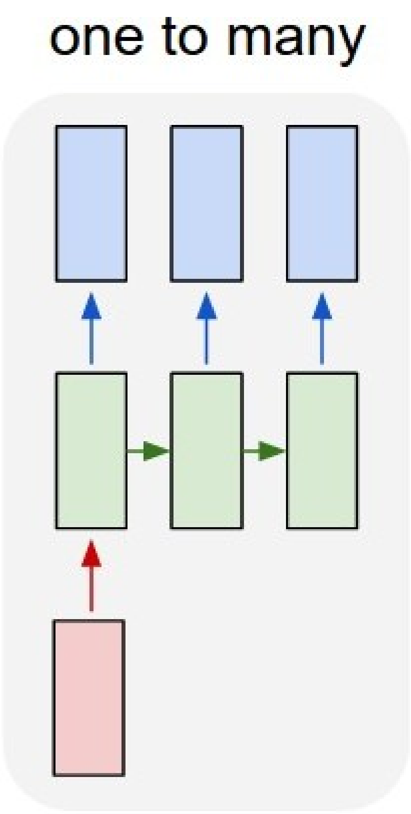


Encoder
- Read sequence
- Predict once

Tasks:
- Sentiment analysis
- Detect age by status
- Week3 homework
- Any text analysis

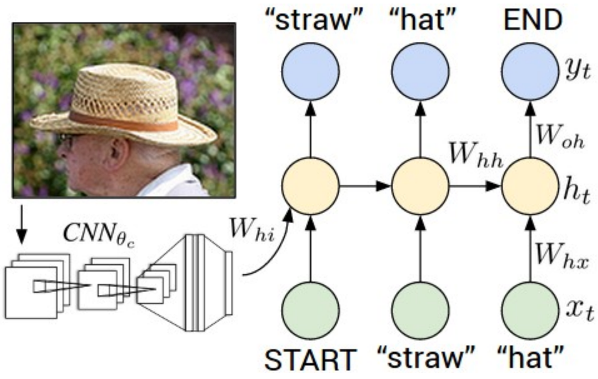
How to implement?
- Take last/max/mean over time

Recurrent Architectures: Decoder



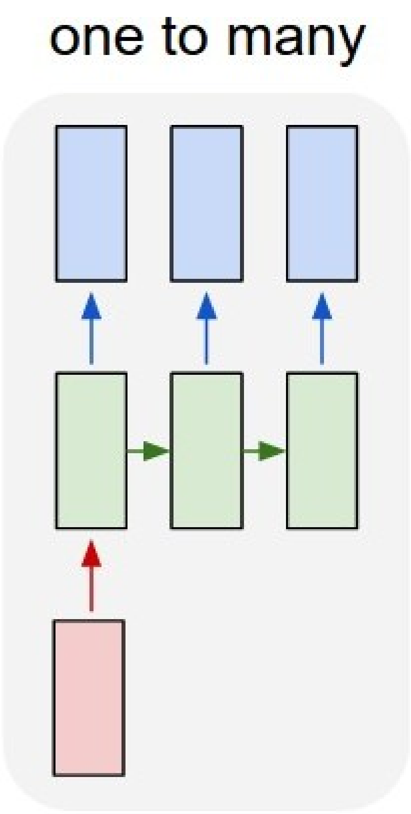
Decoder
- Take one state
- Generate sequence

Tasks:
- Image captioning



How to implement?
- **?!**

Recurrent Architectures: Decoder

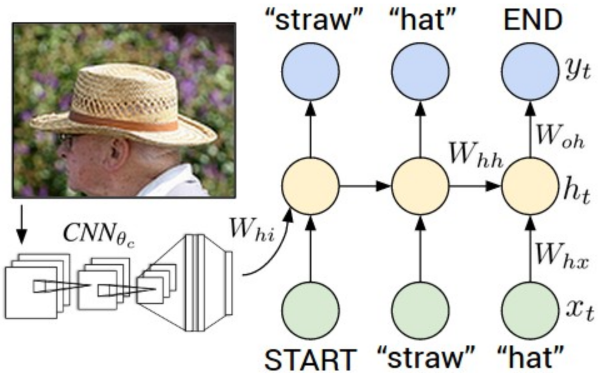


Decoder

- Take one state
- Generate sequence

Tasks:

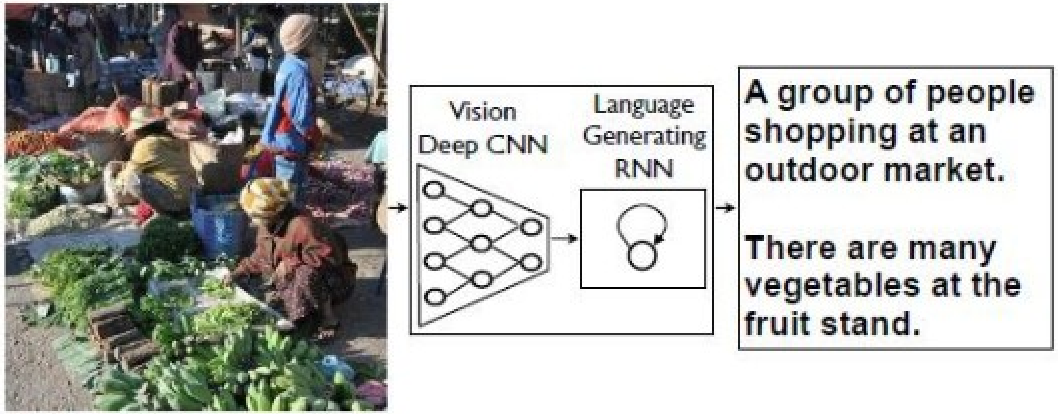
- Image captioning



How to implement?

- First state init (instead of zeros)
- Input at each tick

Image captioning



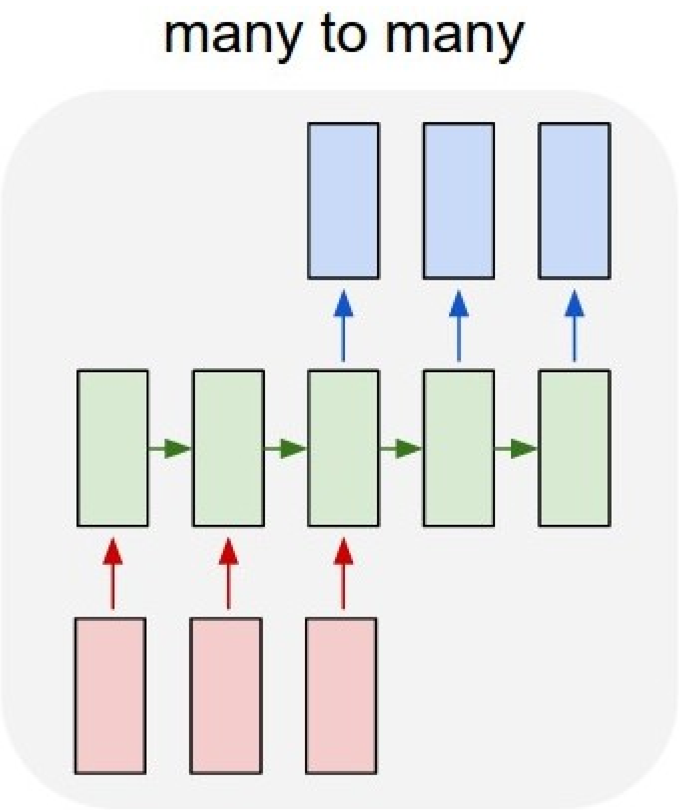
- Demo - http://stanford.io/2esMxOq
- Upload your image - http://bit.ly/2eAoueP

Seq2seq

How do we convert sequence to sequence of different kind/without time synchronization?

Example: **Machine translation**

Seq2seq

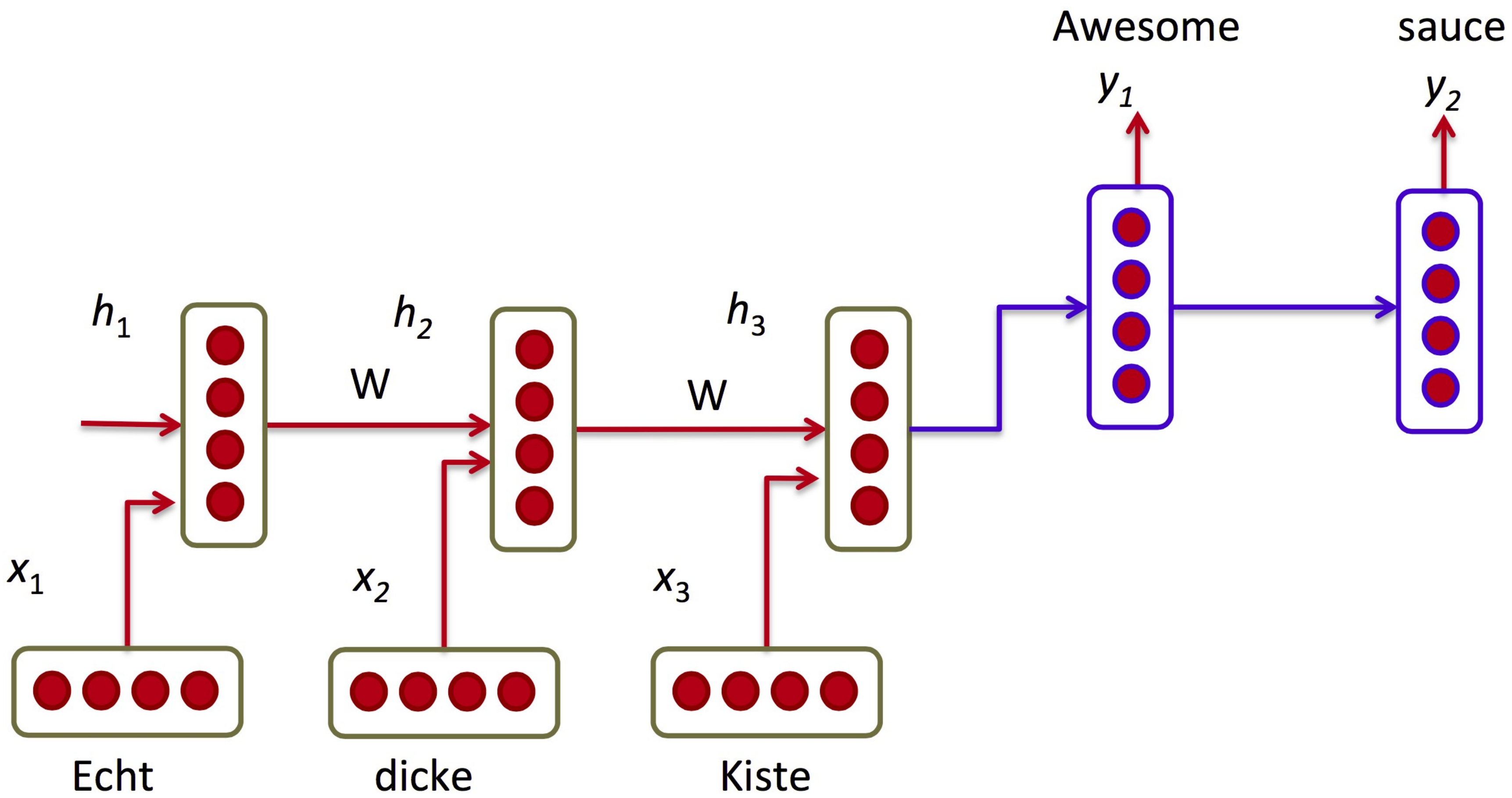


Idea:

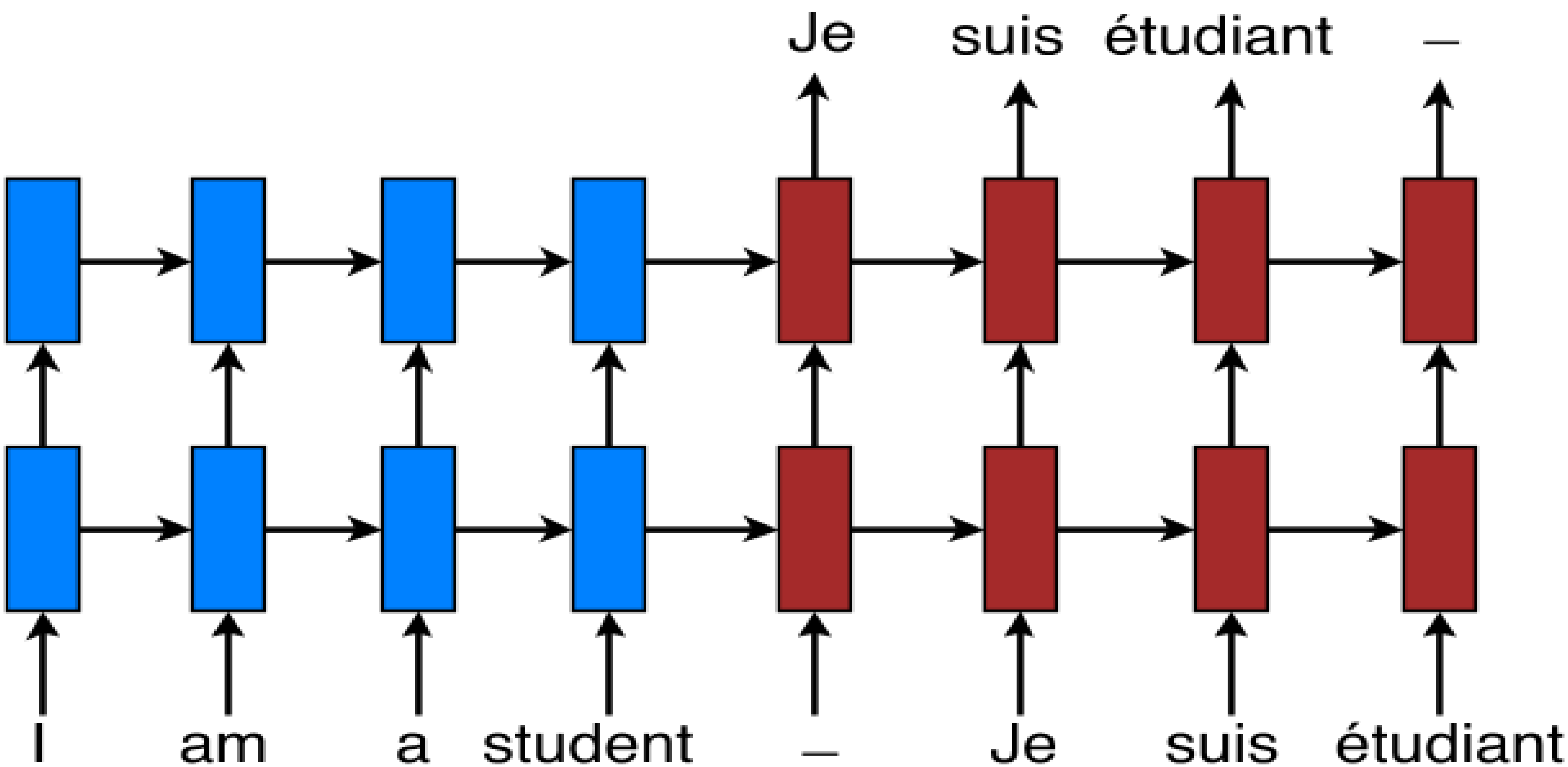
- first read (encode) the sequence
- then generate new one out of the encoded vector

How to implement that?

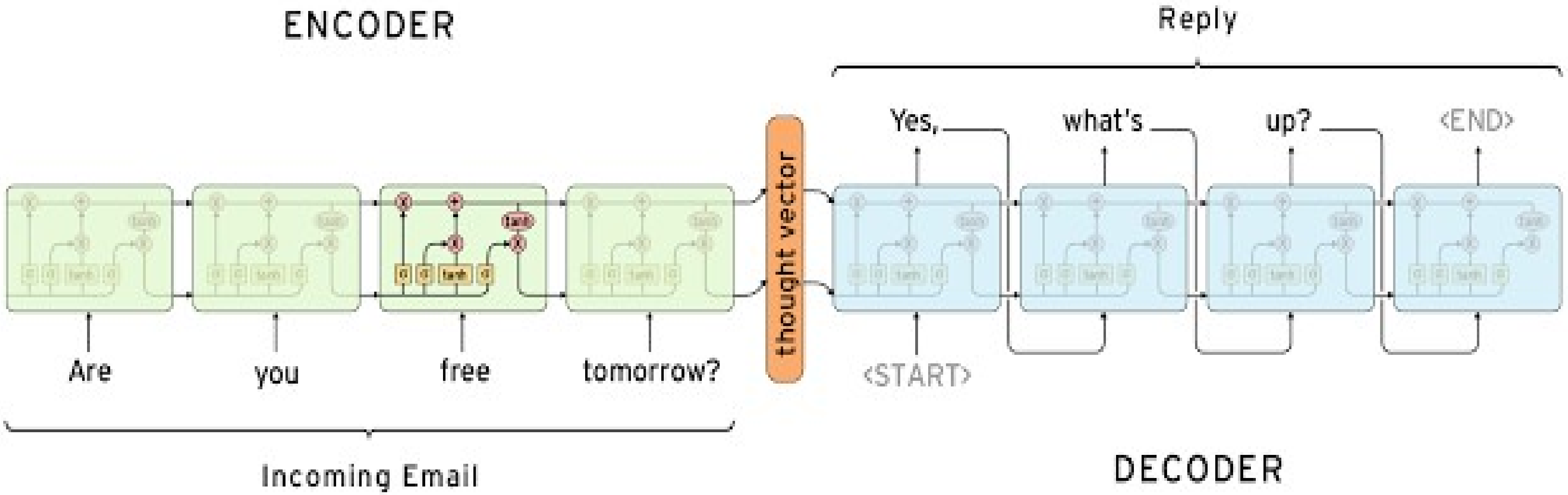
Seq2seq: encoder-decoder



Seq2seq: Machine translation

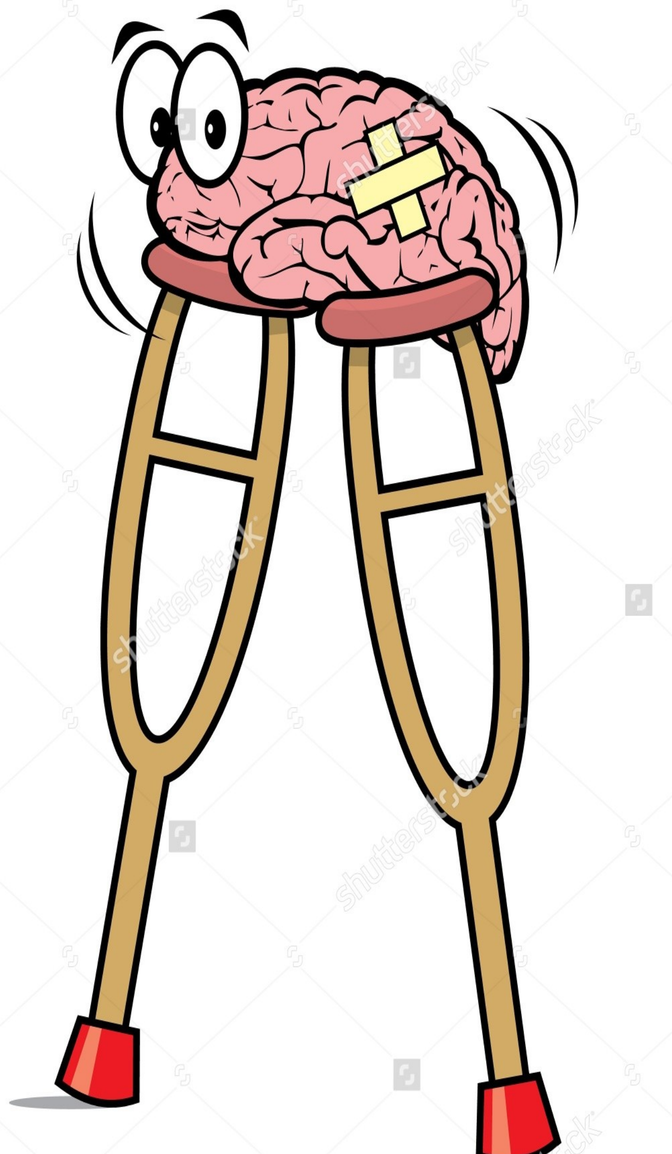


Seq2seq: Conversation model

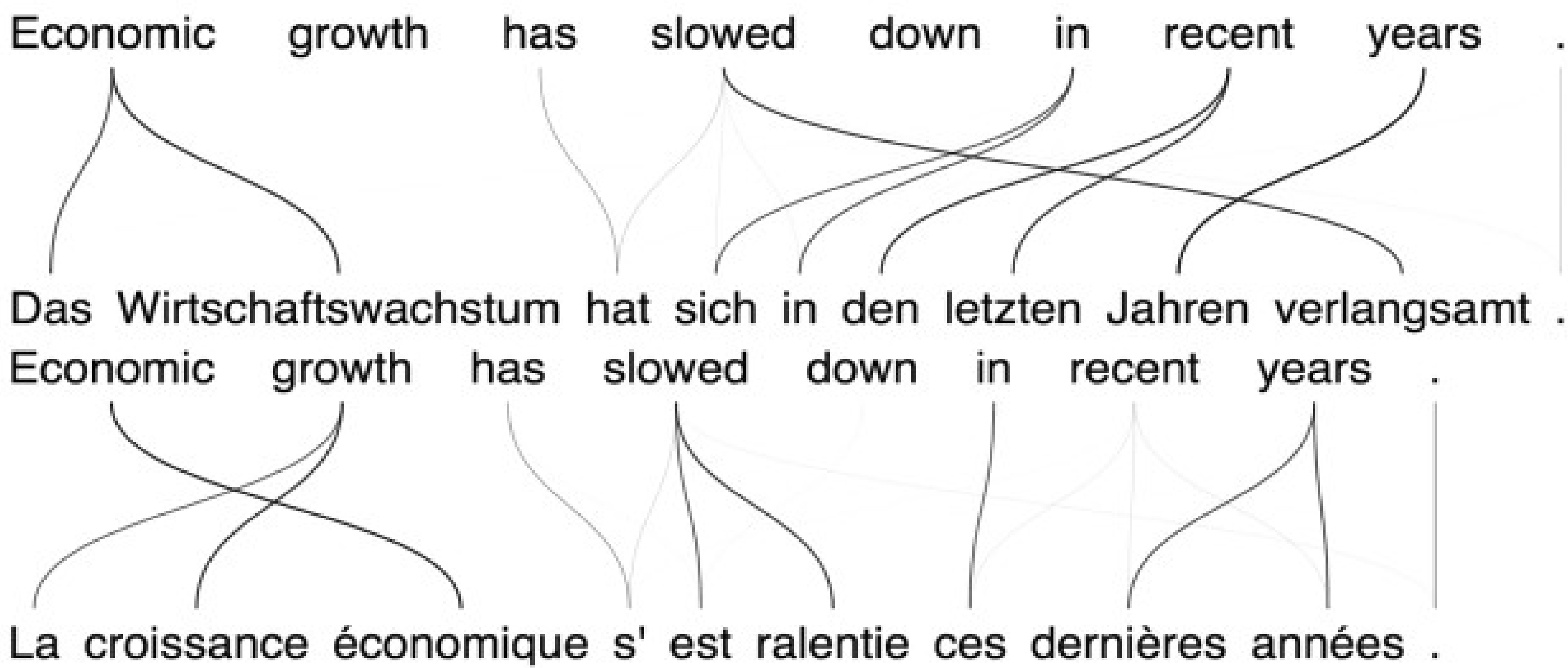


Exactly the same

Augmentations

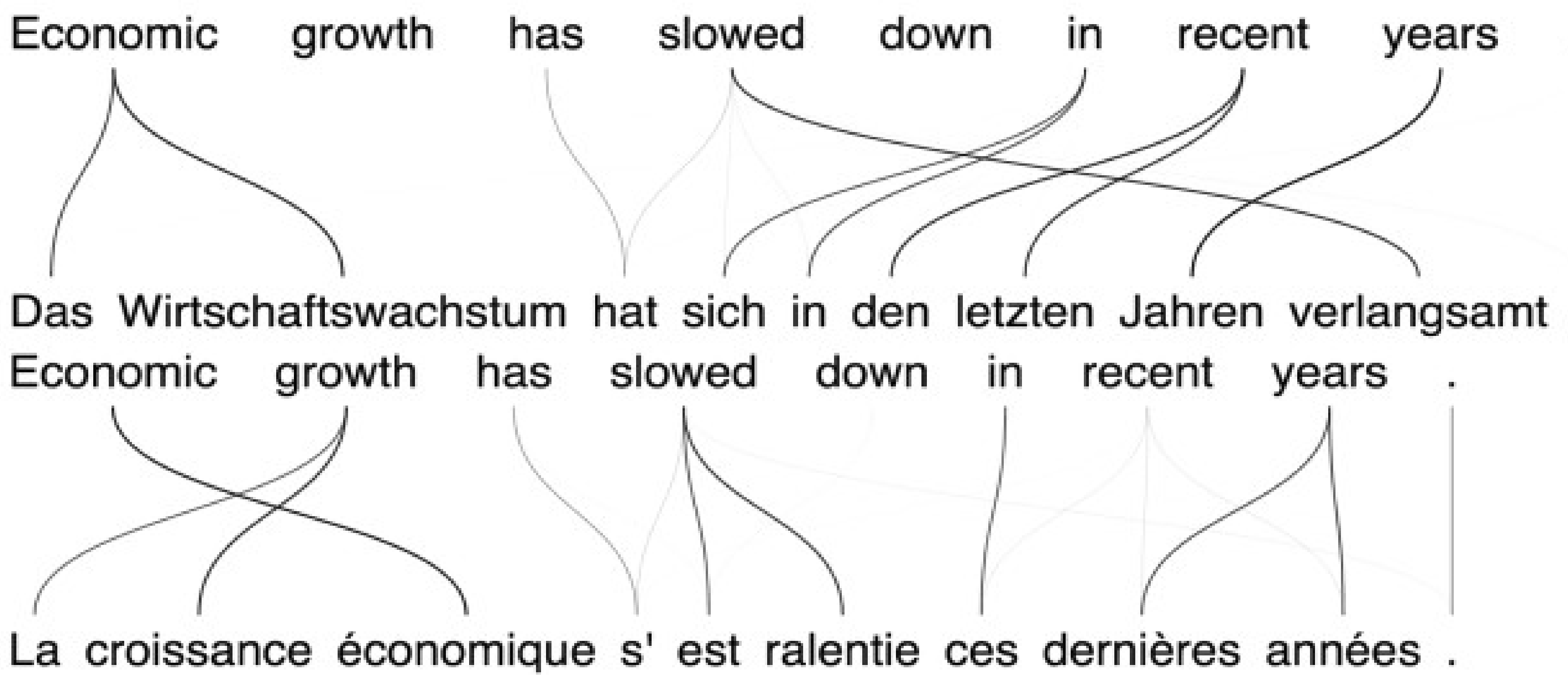


Back to machine translation



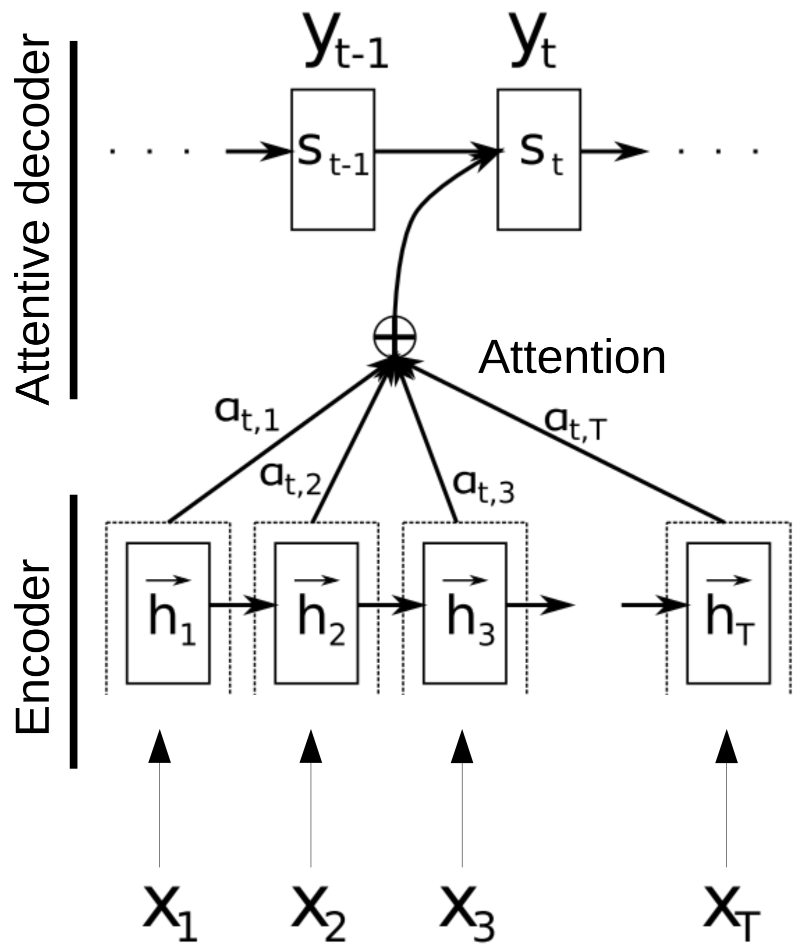
Idea: parallel sentences have corresponding words aligned with language syntax rules

Back to machine translation



Can we learn to **focus** on the correct source word when translating the sentence?

Attention



Idea: on every tick, predict which word do you want to see.

$$P(see\, h_i | s_t) = softmax(A_i(s_t))$$

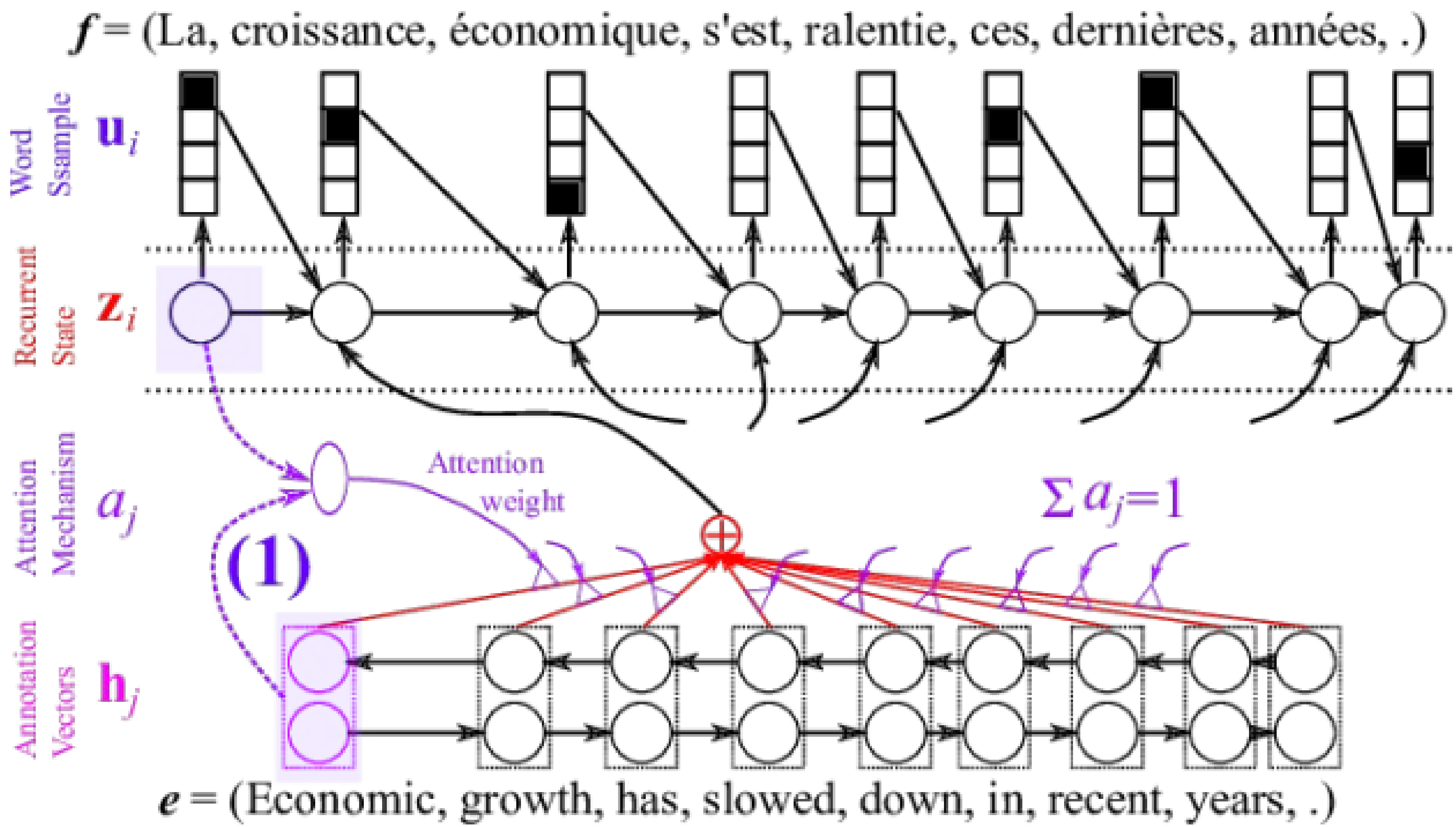
$$Inp_t = \sum_t P(see\, x_i | s_t) \cdot h_i$$

At = another neural net prediction
St = recurrent step (e.g. lstm)

$$s_t(s_{t-1}, y_{t-1}, Inp_{t-1})$$

http://arxiv.org/abs/1409.0473

Back to machine translation



Back to machine translation

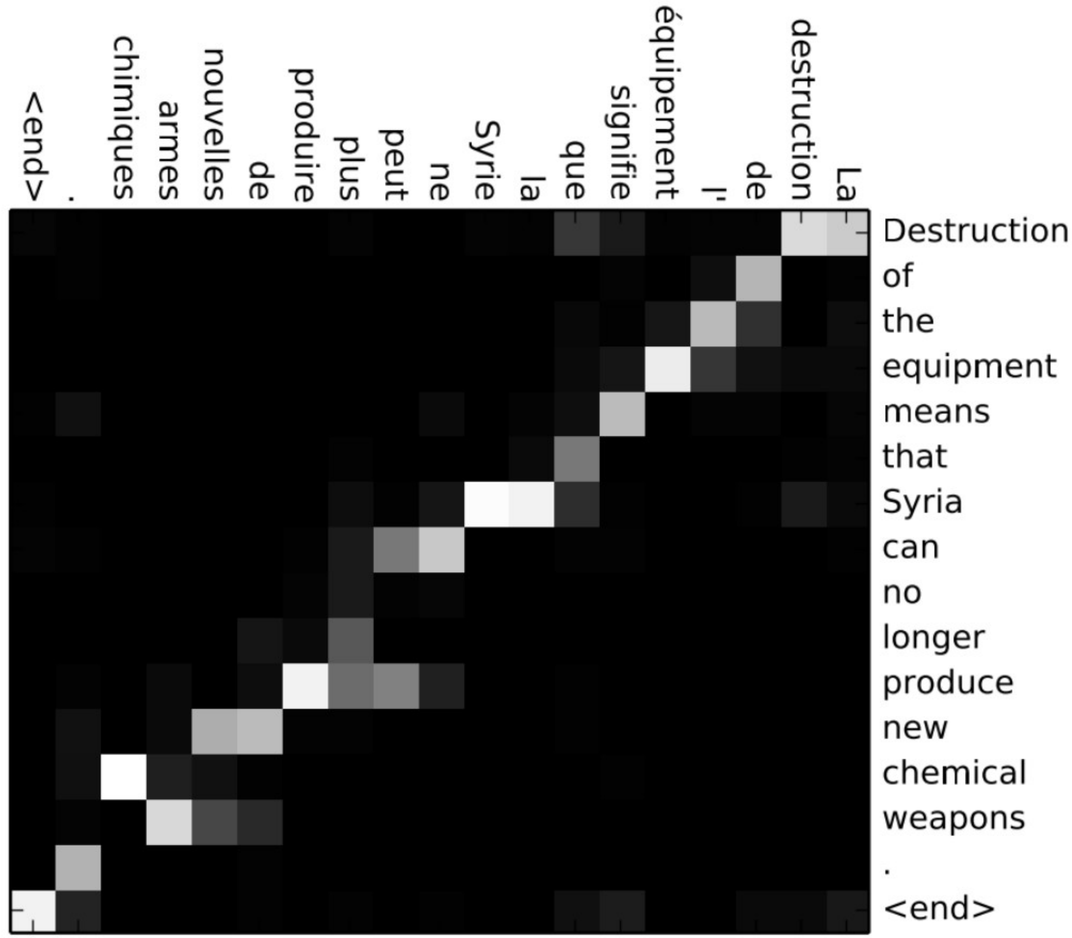
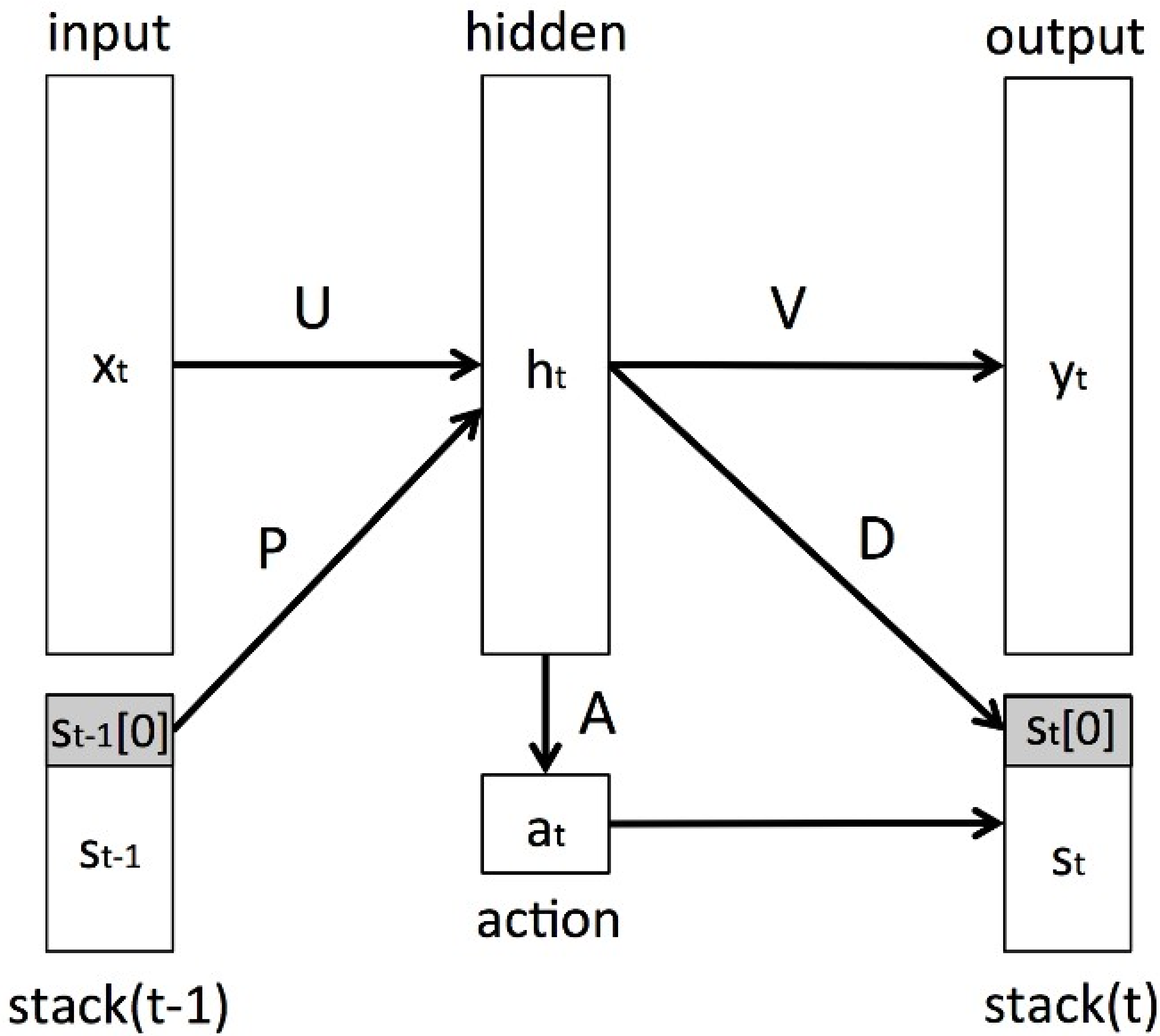
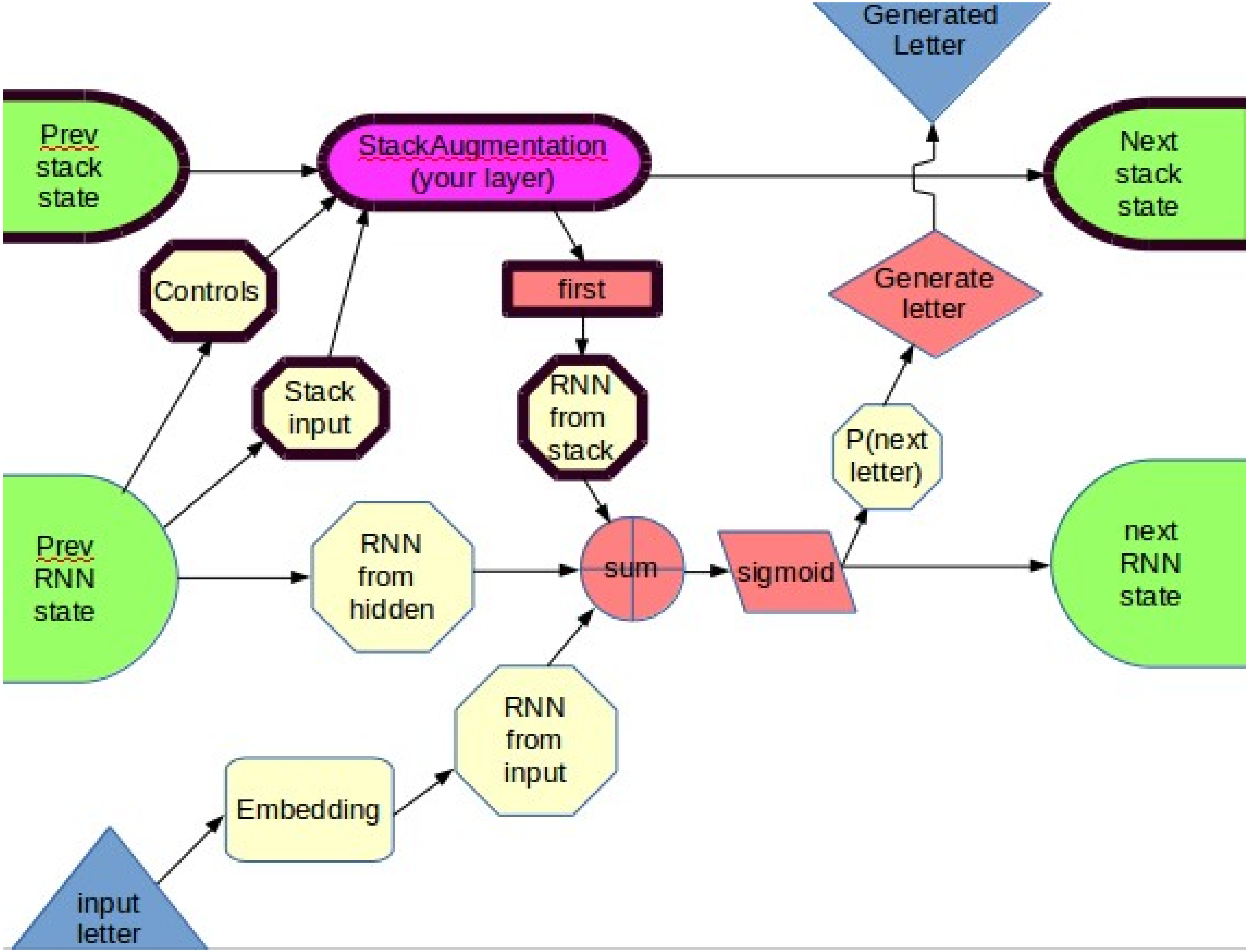


Image: attention vectors: where network "looks" when translating each word

Stack-augmented RNN

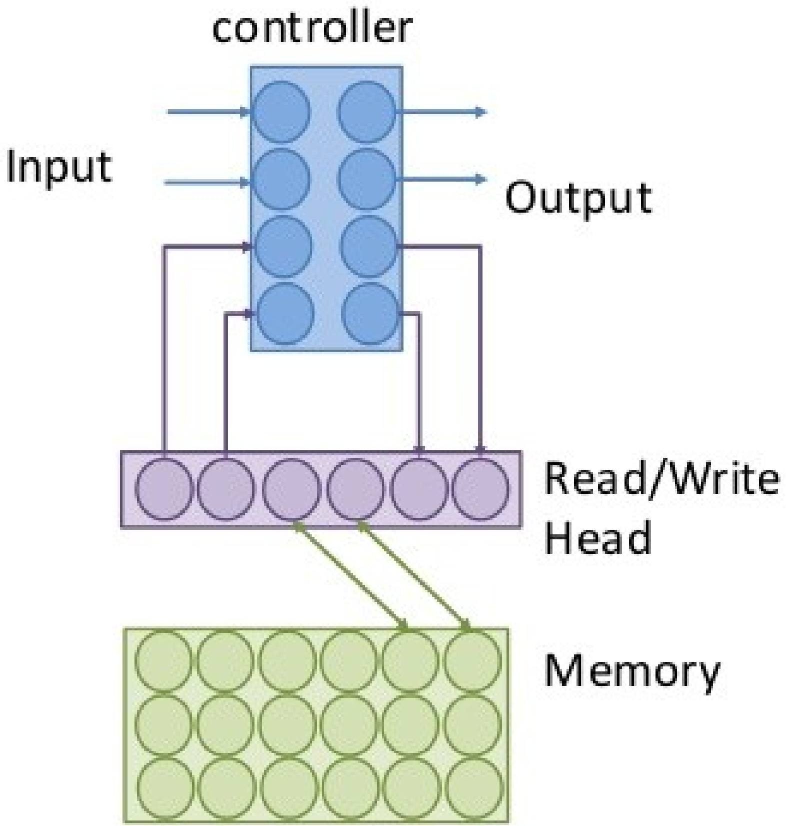


Stack-augmented RNN



Stack-augmented RNN

Neural Turing Machine



Nuff

And now, let's go implement some!