Deep Learning - MAI

Theory - RNNs

Dario Garcia Gasulla
dario.garcia@bsc.es

Context

Vanilla RNNs

Advanced RNNs

RNNs extensions

Context

Dario Garcia Gasulla
dario.garcia@bsc.es

The Sequence

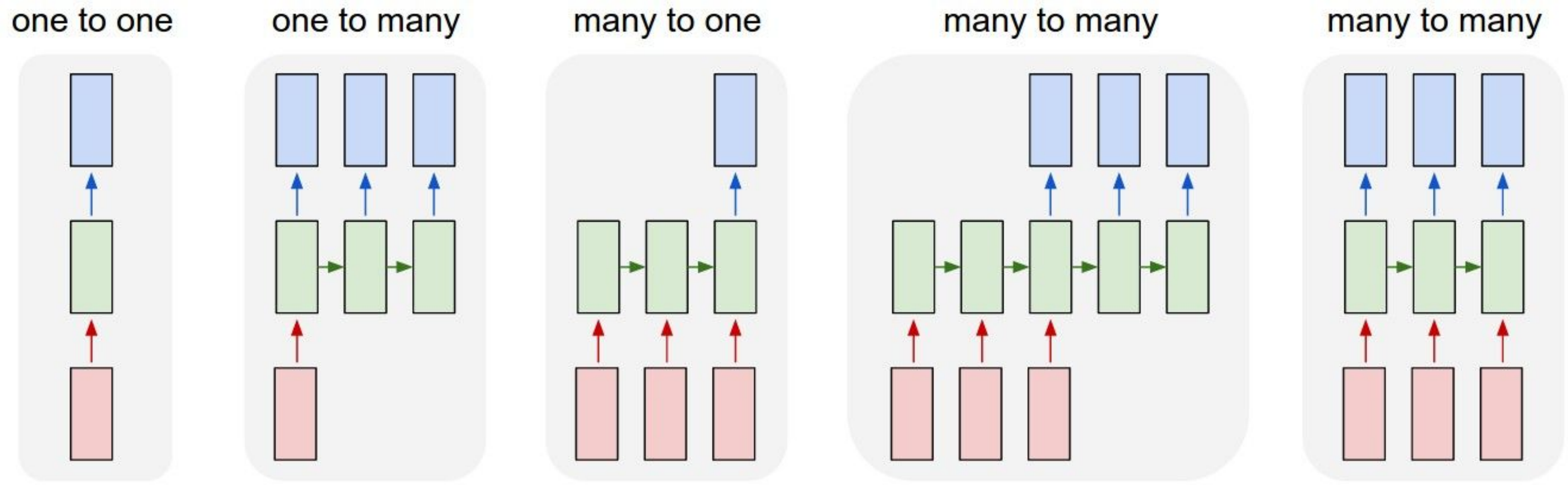
- Fully-connected and CNNs* have fixed inputs and outputs
- What if we have a variable shape input? Or output?
- Given a sequence (Text, video, audio, signal, bioinformatics...)
  - Learn the relations between the symbols that compose it

** CNNs can have a certain degree of variable inputs/outputs thanks to convolution*

What do we want

- Sequences can be processed using traditional methods (e.g., sliding windows), but sequences need to have the same length
- We want to…
  - process sequences of arbitrary length
  - provide different mappings (one to many, many to one, …)

The ways of the sequence



Not a RNN (why?)

Sequence output

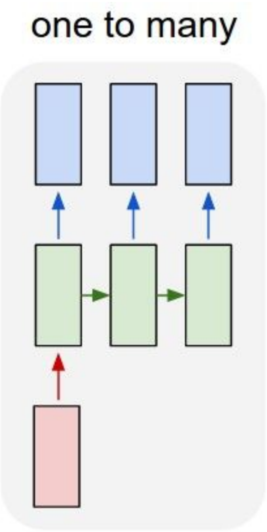
Sequence input

Sequence to sequence

Synced sequence

One to many

- Image captioning



"man in black shirt is playing guitar."

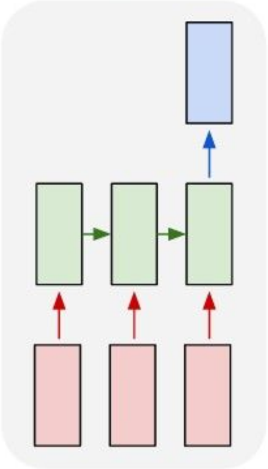
"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

Many to one

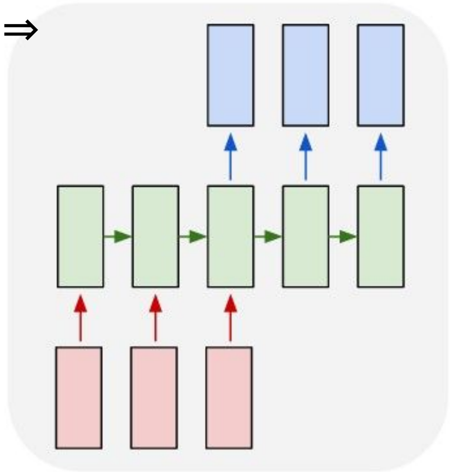
- Sentiment analysis
  - My flight was just delayed, s**t ⇒ Negative
  - Never again BA, thanks for the dreadful flight ⇒ Negative
  - We arrived on time, yeehaaa! ⇒ Positive
  - Another day, another flight ⇒ Neutral
  - Efficient, quick, delightful, always with BA ⇒ Positive

many to one



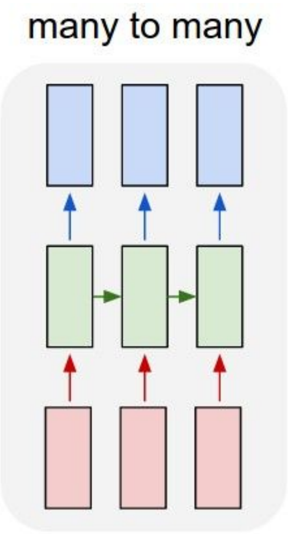
Sequence to Sequence

- Automatic translation
  - [How, many, programmers, for, changing, a,lightbulb,?] ⇒
  - [Wie, viele, Programmierer, zum, Wechseln, einer,Gl ühbirne,?] ⇒
  - [Combien, de, programmeurs, pour, changer, une,ampoule,?] ⇒
  - [¿,Cuantos, programadores, para, cambiar,una,bombilla,?] ⇒
  - [Zenbat, bonbilla, bat, aldatzeko,programatzaileak,?]



Synced Sequence

- Frame classification

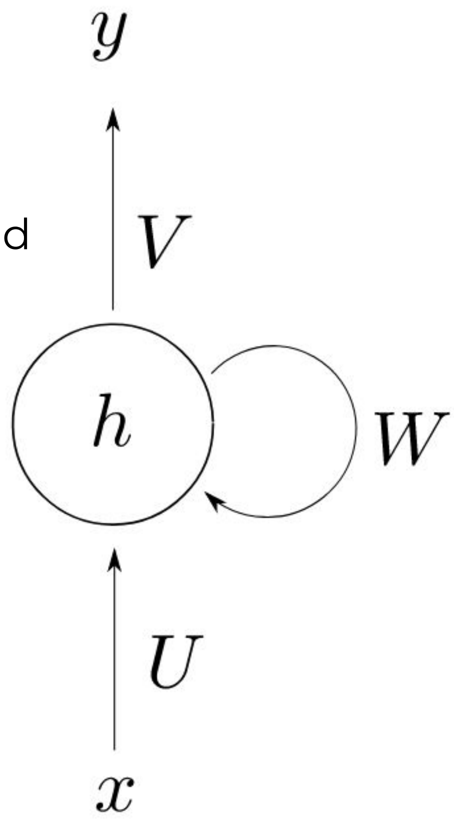


Vanilla RNNs

Dario Garcia Gasulla
dario.garcia@bsc.es

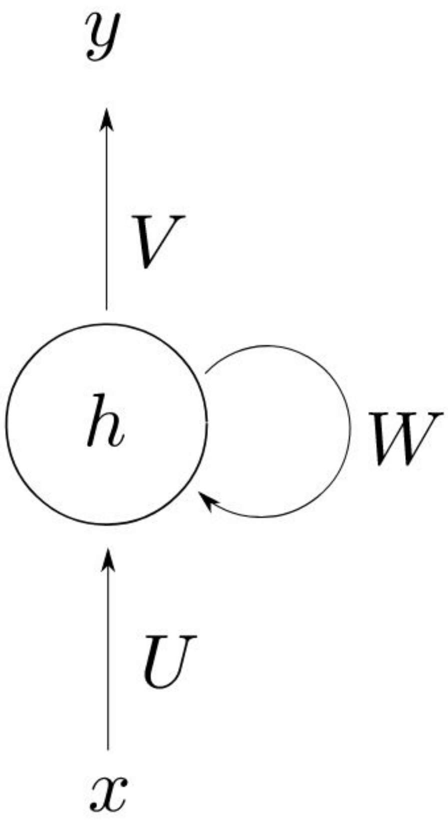
What makes a RNN?

- Recurrent Neural Networks are feed-forward networks with edges that span adjacent time steps (recurrent edges)
- On each step, a neuron receives inputs from data (as usual) and from previous time steps (history)
- In other words, previous time steps can influence future time steps
- RNNs are universal function approximators (Turing Complete)

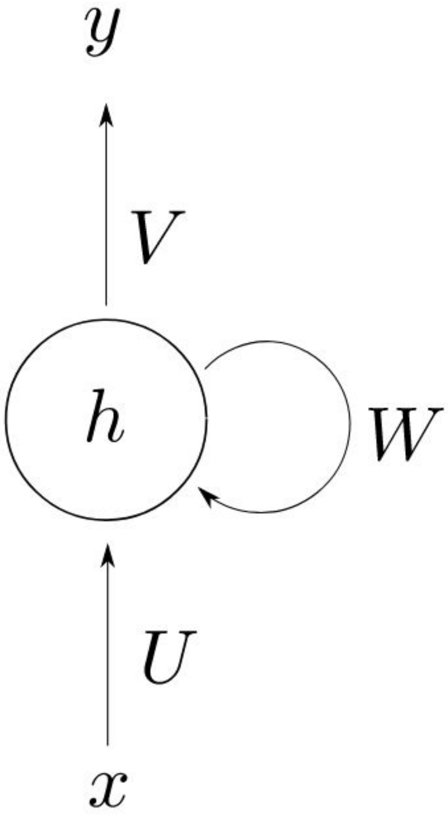


Who is who

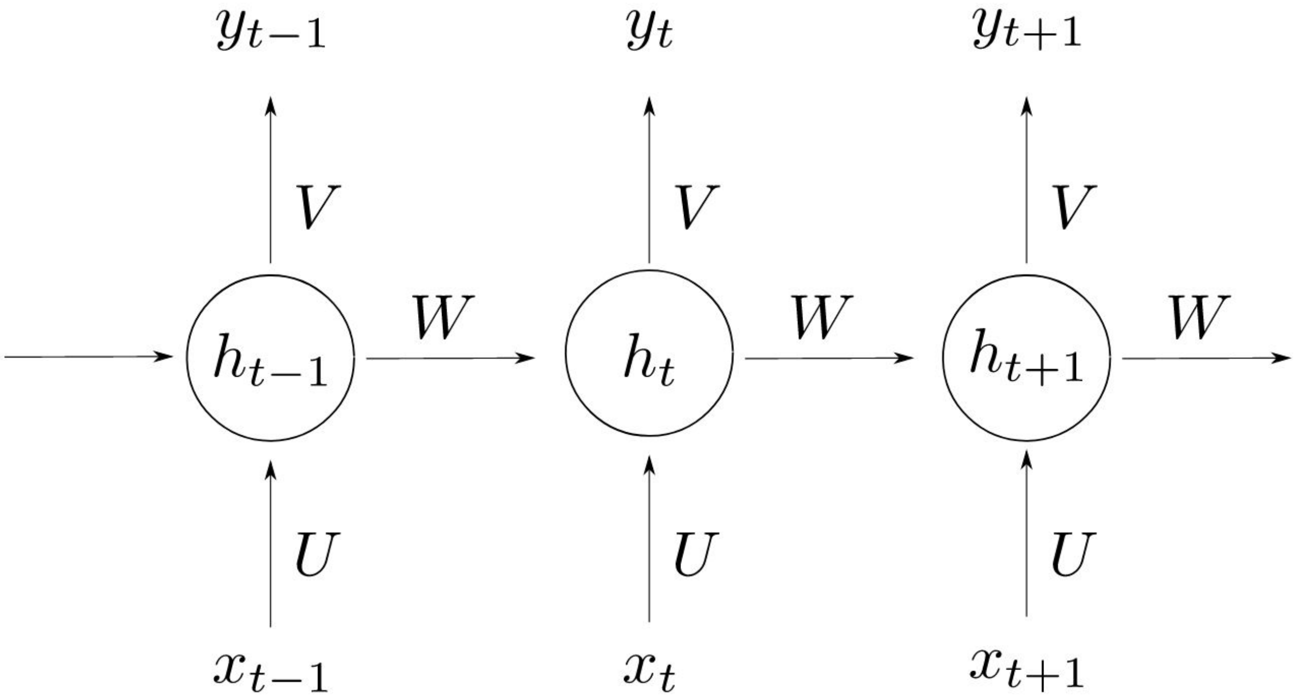
- RNNs Input (**x**) is a vector of values for time t
- The hidden node (**h**) stores the state
- Weights are shared through time (input independent!)
- Each step the computation uses the previous step
  - $h^{(t+1)}=f(h^{(t)},x_{t+1};\theta) = f(f(h^{(t-1)},x_t;\theta),x_{t+1};\theta) = \cdots$
- RNN are a deep network that stacks layers through time



Before unrolling

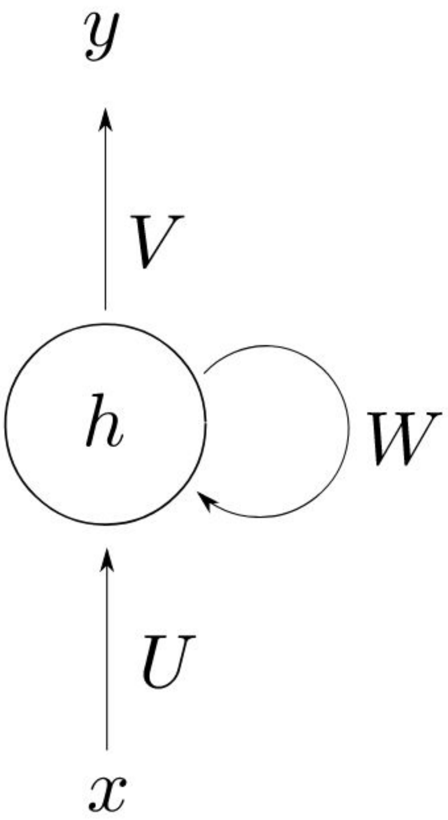


After unrolling



The computation

$$
\begin{aligned}
a^{(t)} &= b + W \cdot h^{(t-1)} + U \cdot x^{(t)} \\
h^{(t)} &= tanh(a^{(t)}) \\
y^{(t)} &= c + V \cdot h^{(t)}
\end{aligned}
$$



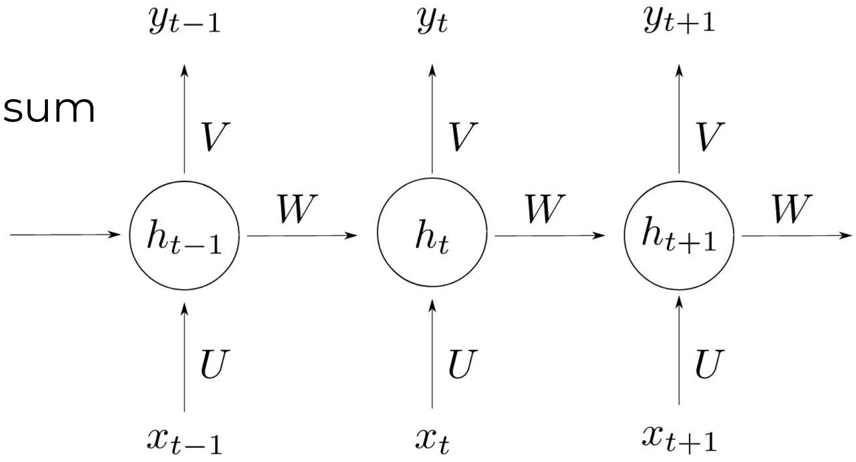
Practical Tips I

The power of RNNs lies in the unrolling, not in depth

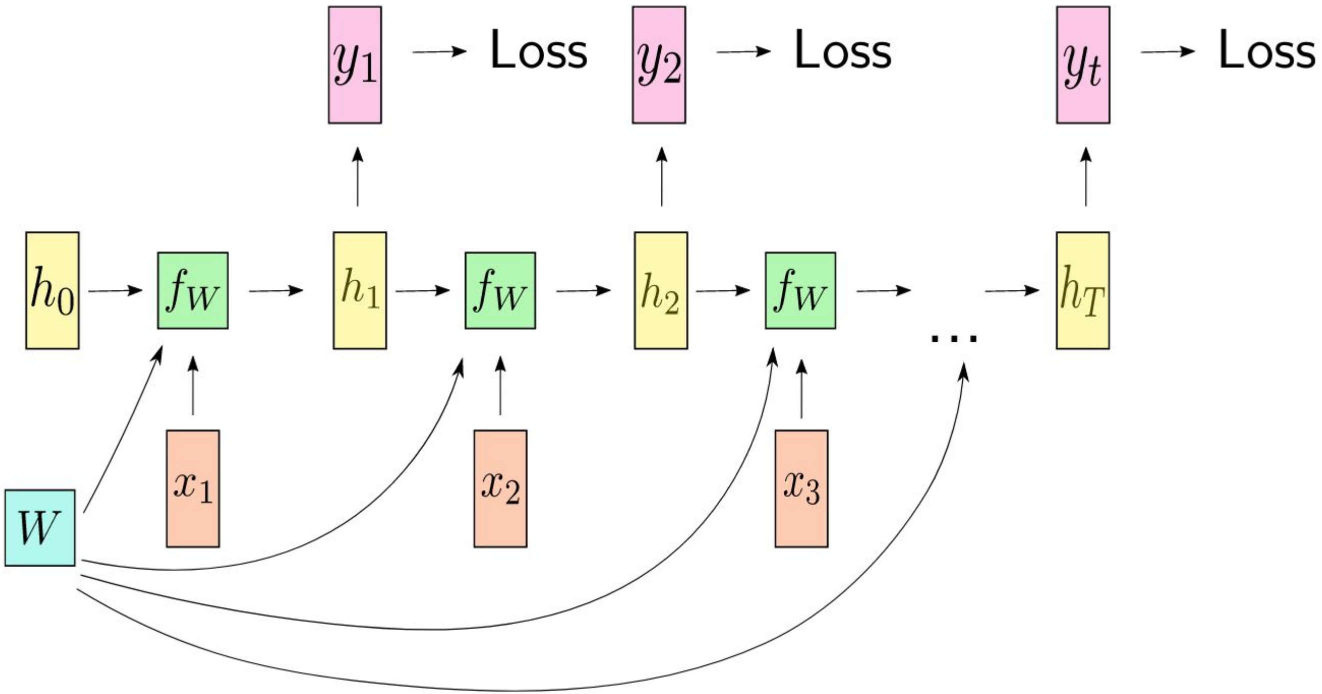
- RNNs with many layers are very expensive to compute
- Plus, the real challenge is in the memory
- Activation functions
    - Tanh is better than sigmoid
    - ReLU is also popular
- When working with text, inputs are most frequently *word embeddings*

How to train RNNs

- Backprop + SGD on the unfolded sequence
  - Compute activations and gradient
- Backpropagation Through Time (BPTT) by sum
- Input is limited in time to limit cost
  - Truncated BPTT
  - Influence limited to a time horizon



Combining losses



Training issues

- Sharing weights, with longer sequences, easily yields
  - Vanishing gradient (favours close by patterns wrt distant ones)
  - Exploding gradient
- Clipping gradients to prevent exploding
  - Scale gradient if the norm is above a threshold
- Vanishing gradients, next

Key features of RNNs

- Can process any length of input
  - For fixed sized inputs, consider other options
- Implements memory by design
- Model complexity is independent of input length
- All inputs processed with the same weights - reuse knowledge

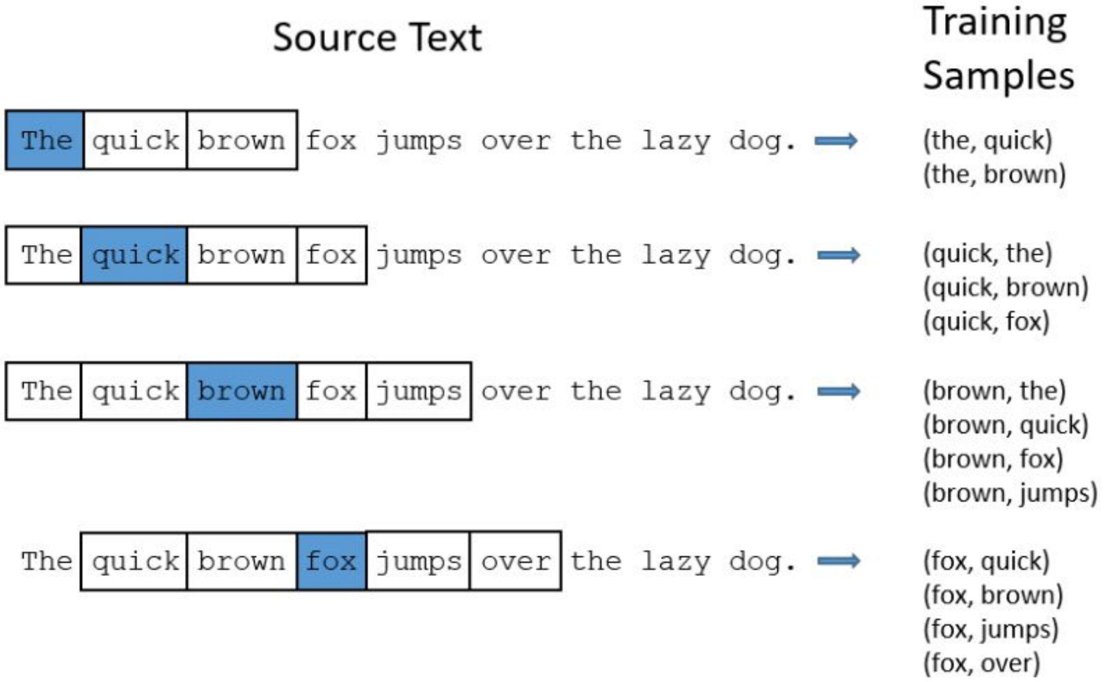
RNNs inputs

- Word embeddings!
  - Language models
  - Unsupervised learning
  - Shallow NN
- word2vec vs glove (nobody cares anymore)
  - Expensive to train

[51,52]

Word embedding task

- Words are defined by their context
- Endless source of training data
- Use a sliding window of fixed length

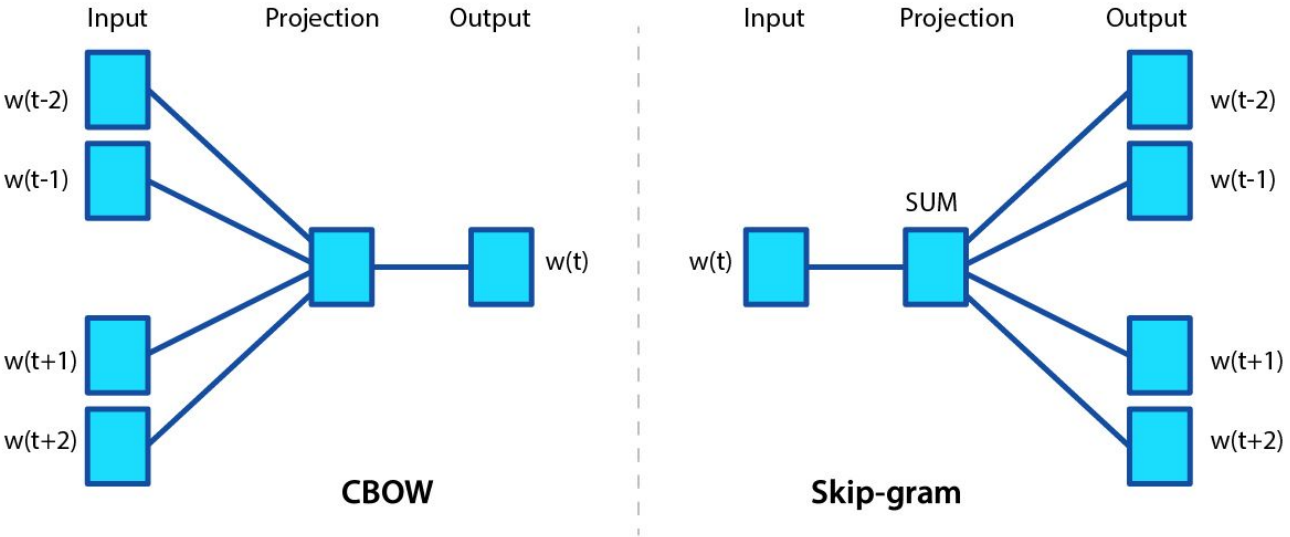


Word embedding models

Forget about the output!
Give me the intermediate representations

- Predict word given context

Predict context given word



- Faster and slightly better for frequent words
- Good for little training data and rare word representation

Word embedding properties

- Word similarity (cosine distance among vectors)
- Regularities
  - Add & Substract
- So much bias...
- Go play

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

  - https://projector.tensorflow.org/ https://ronxin.github.io/wevi/

Advanced RNNs

Dario Garcia Gasulla
dario.garcia@bsc.es

The limitations

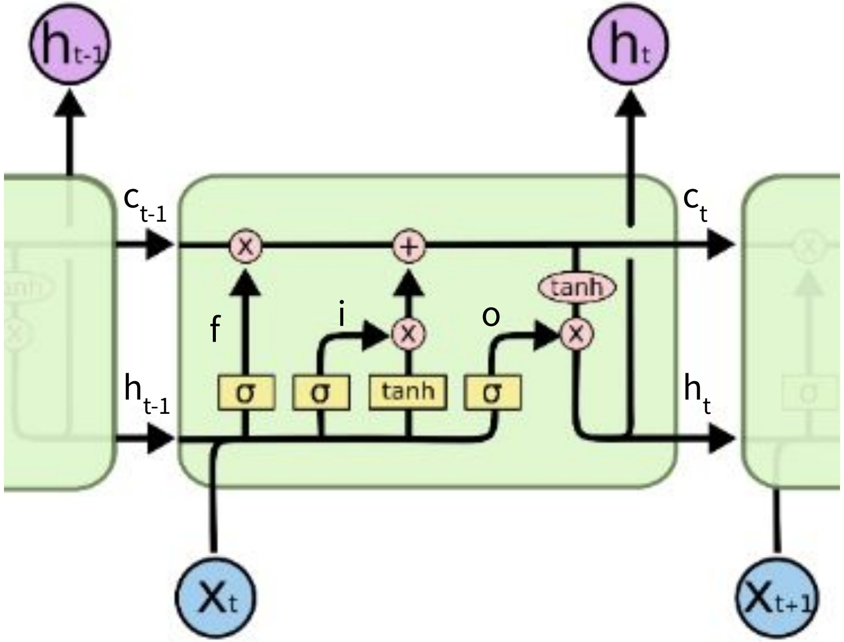
- RNNs keep multiplying the same weight matrix over and over, which makes it error prone
- Keeping long term relations are hard for RNN
- The state has to encode both short and long term relations, which gets complicated as input sequences grow
- What if we make the state more powerful?

LSTM

- Long-Short Term Memory
  - In addition to the hidden state, add a cell state
  - Hidden state stores short-term info (large updates)
  - Cell state stores long-term info (small updates)
  - Include gate operators to erase, write and read from the cell
- Gates allow regulating the cell state, deciding the operation (e/r/w), its target (cell state segment) and magnitude (gate in range [0,1]), based on context

Inside a LSTM

- Forget gate: What is kept in the cell
  - $f_t = \boldsymbol{\sigma}(W_f \cdot [h_{t-1}, x_t])$
- Input gate: What is added to the cell
  - $i_t = \boldsymbol{\sigma}(W_i \cdot [h_{t-1}, x_t])$
  - $\hat{c}_t = \mathbf{tanh}(W_c \cdot [h_{t-1}, x_t])$ (new content)
- Output gate: What is passed to the hidden
  - $o_t = \boldsymbol{\sigma}(W_o \cdot [h_{t-1}, x_t])$
  - $c_t = f_t c_{t-1} + i_t \times \hat{c}_t$ (new cell state)

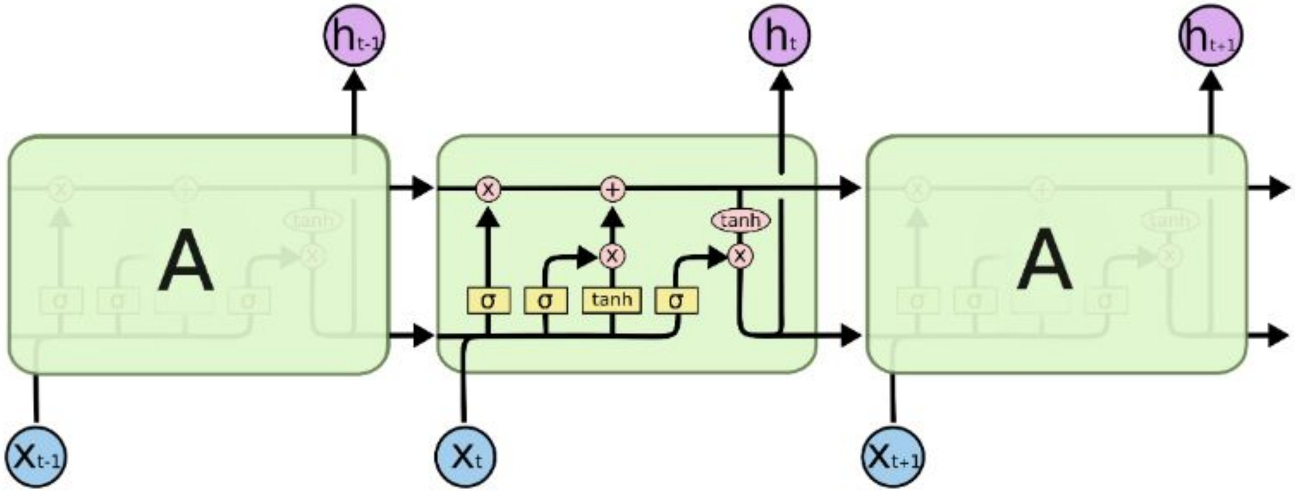


$h_t = o_t \times \mathbf{tanh}(c_t)$ (new hidden state)

Why do LSTM work

- If forget gate is set to 0 (remember everything), long-term relations are always kept
- It includes a sort of short-cut, as long as you do not forget everything, gradient will flow!*

* No more W exp, but sigmoid can still vanish the gradient

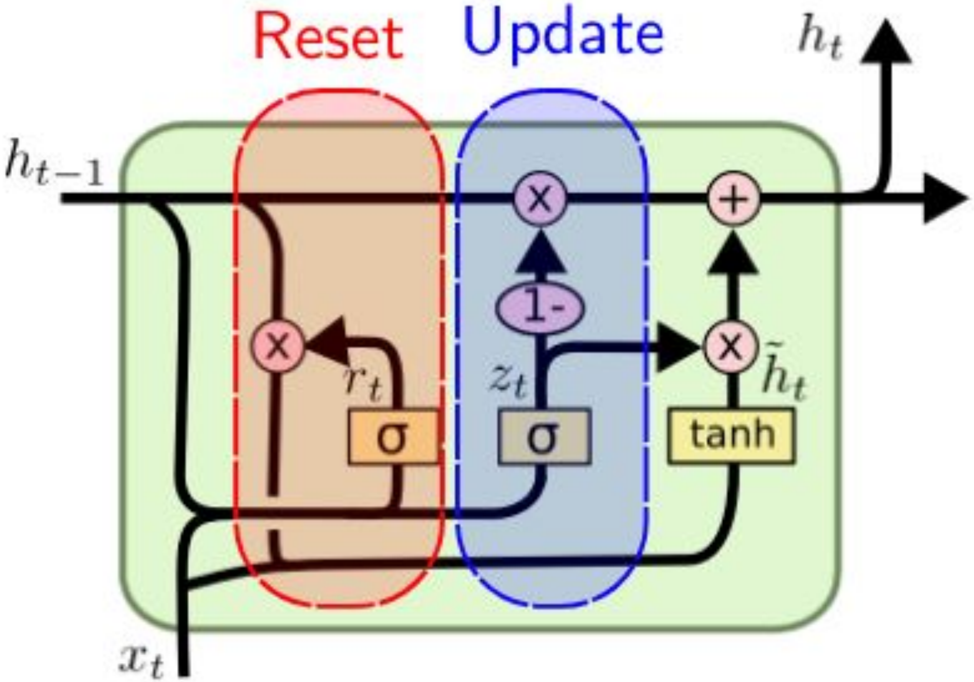


Gated Recurrent Units

- A simplified version of LSTMs (thanks!)
  - No cell state
  - Update gate: What in the hidden state is updated/left as it is
    - LSTMs forget gate + input gate
  - Reset gate: Which part of the *previous* hidden state are used
    - Close to 1: Previous state has more relevance
    - Close to 0: New state has more relevance

Inside a GRU

- Update gate: How to change hidden state
  - $z_t = \boldsymbol{\sigma}(W_z \cdot [h_{t-1}, x_t])$
- Reset gate: How to combine
  - $r_t = \boldsymbol{\sigma}(W_r \cdot [h_{t-1}, x_t])$
- New hidden state content
  - $\hat{h}_t = \textbf{tanh}(W_h \cdot [r_t \times h_{t-1}, x_t])$
- Hidden state output
  - $h_t = (1-z_t) \times h_{t-1} + z_t \times \hat{h}_t$



Vanilla RNN vs LSTM vs GRU

- Of all RNN variants, LSTMs and GRUs are the most widely used
  - Vanilla falls short in complex tasks, lacking long memory capacity
- Main difference between LSTM and GRUs: Complexity
  - Training GRUs is faster and includes less parameters
- Performance-wise, there are no consistent results

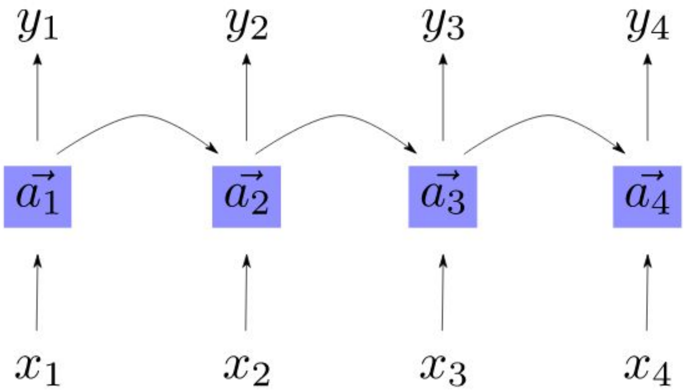
[45,46]

RNNs Extensions

Dario Garcia Gasulla
dario.garcia@bsc.es

Bidirectional RNNs

- Hidden states encode *past* states to influence current state

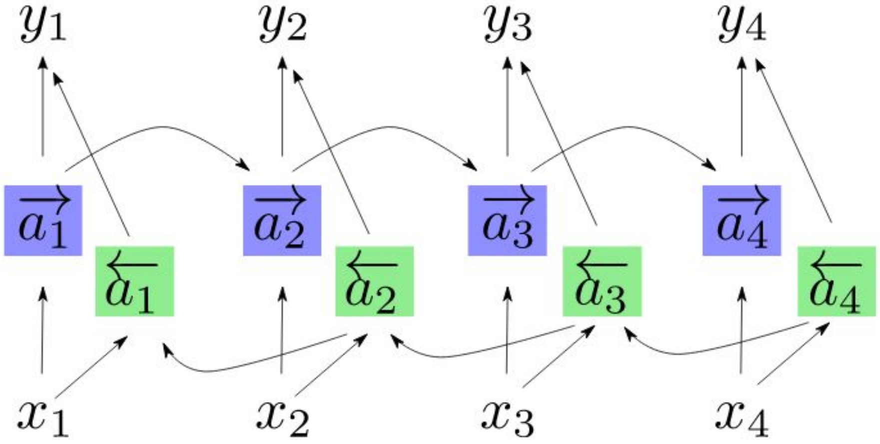


- What about future states? What if read from end to start?
  - PoS tagging, machine translation, speech/handwritting recognition

Bidirectional RNNs

- “Reading” in both directions at the same time

- Two RNNs, one in each direction, with different weights, concatenating their outputs

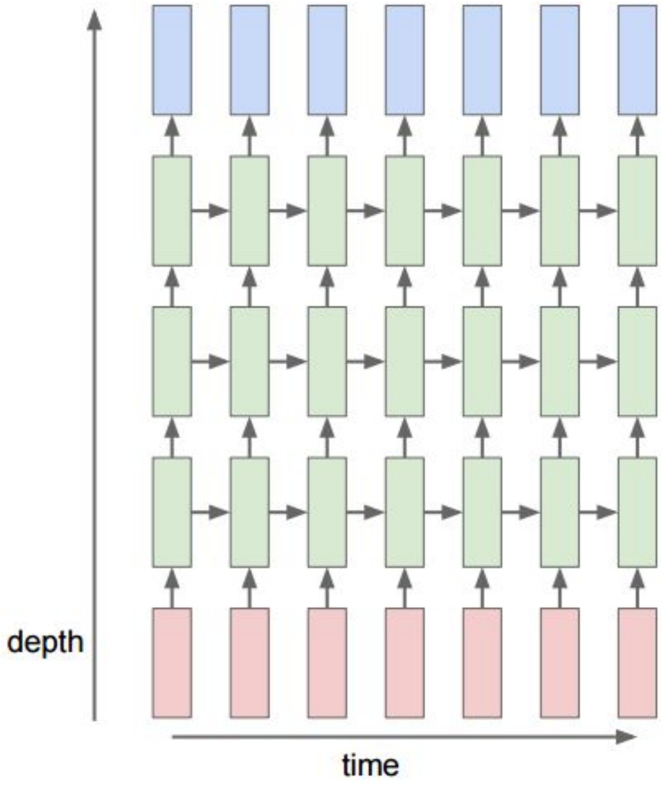


Training Bidirectional RNNs

- Both RNNs trained concurrently, but dependencies must be respected
  - Forward: Compute output of both RNNs and combine step by step
  - Backward: Compute gradient of both RNNs and combine step by step
- If possible (complete input sequence available), use bidirectionality

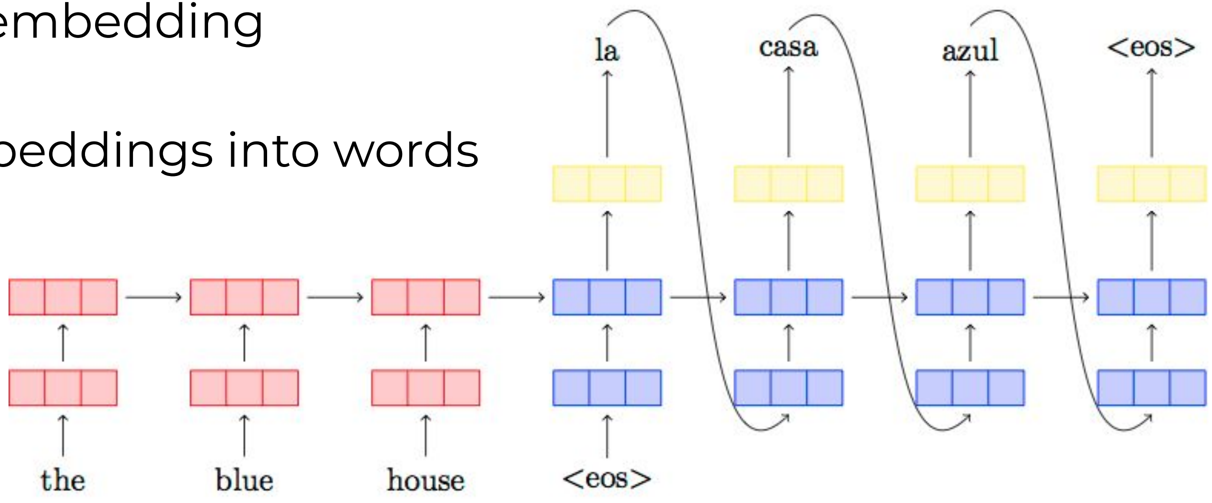
Multi-layer RNNs

- So far, only one layer used
  - Depth is provided by unrolling.
- Multi-layer RNNs
  - Stacking layers (rarely more than 4)
  - Provide extra abstraction capacity
  - A computational nightmare



Encoder-Decoder RNNs (seq2seq)

- Sequence to sequence of different length (Neural Machine Translation)
- One RNN encodes words within their context
- Generates a sentence embedding
- One RNN decodes embeddings into words
- Start and end tokens



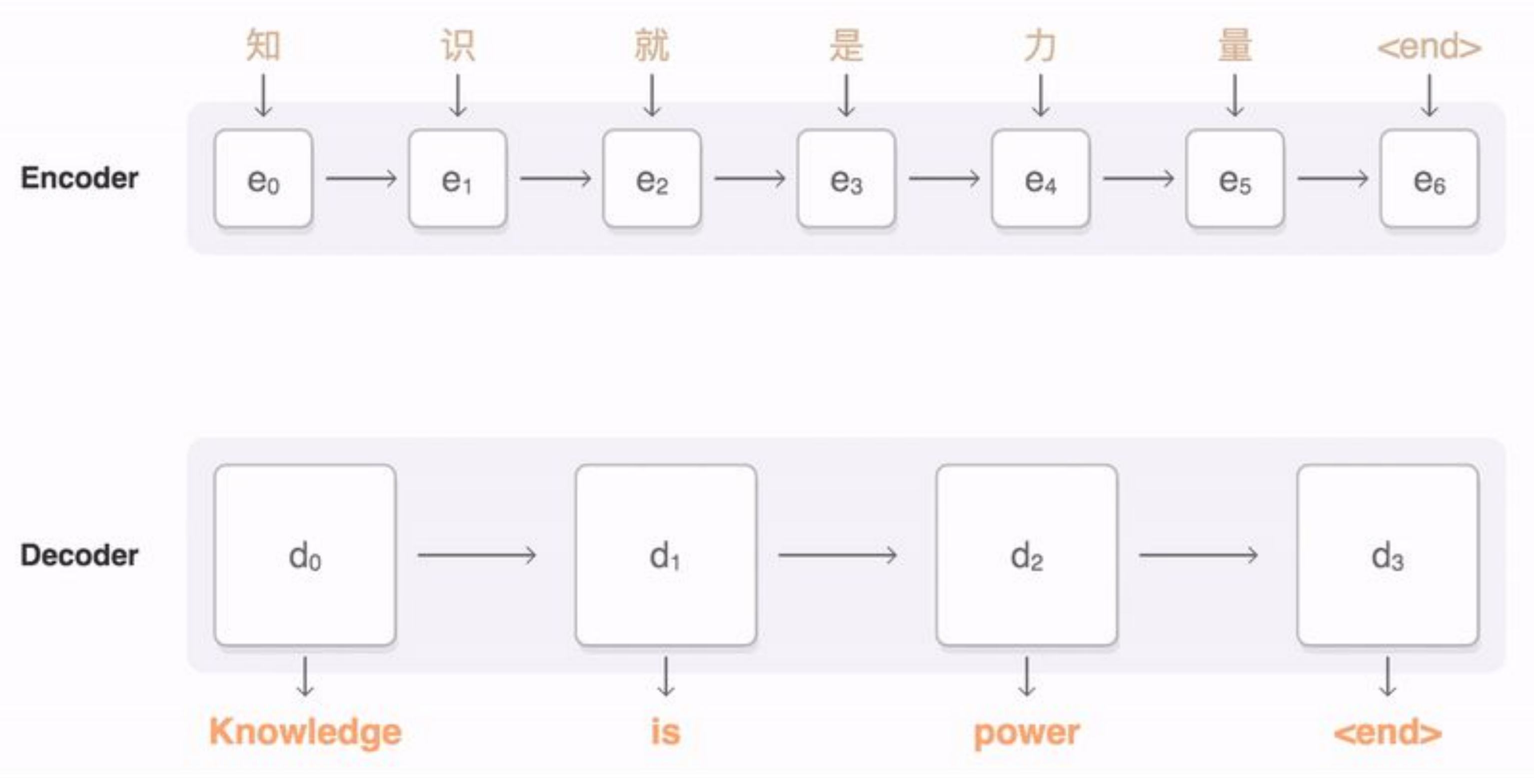
Encoder-Decoder practical details

- Applications
  - Automatic language translation
  - Dialogue generation
  - Document summarization
  - Automatic response generation
  - Input parsing
- Training
  - Each decoding step generates one loss
  - Negative log-likelihood of the true outcome at that point
  - Average all losses at each step
  - Decoder feedback replaced by true outcome

From Encoder-Decoder to Attention

- seq2seq limitations
  - Full sentence into a fixed-sized, unique embedding (bottleneck)
  - Different parts of the decoder focus on different parts of the input
  - Greedy decoding: Carrying on errors
  - Beam search decod.: Keep top k branches, find most probable path
- Solution: Attention
  - Let each decoder step decide which part of the input use

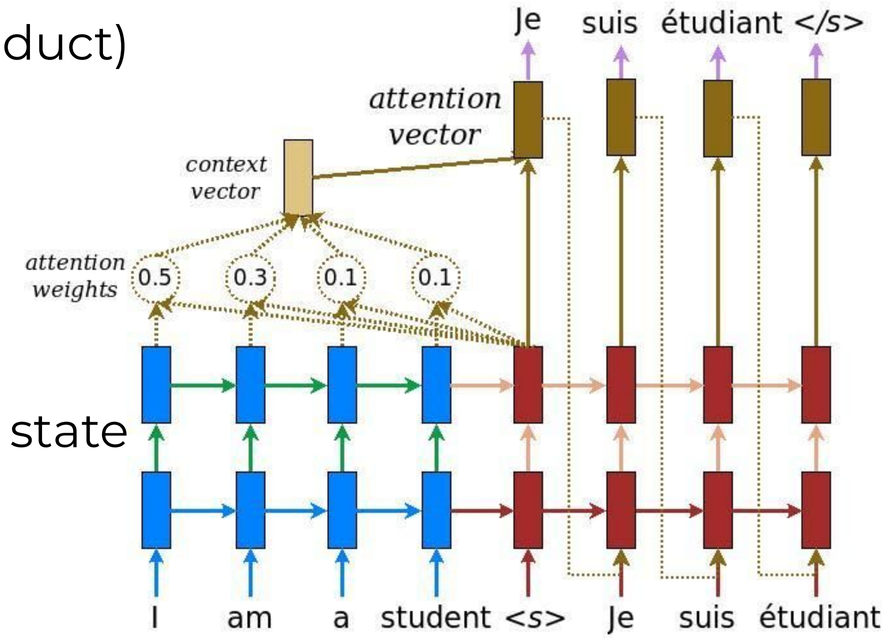
Attention overview



[50]

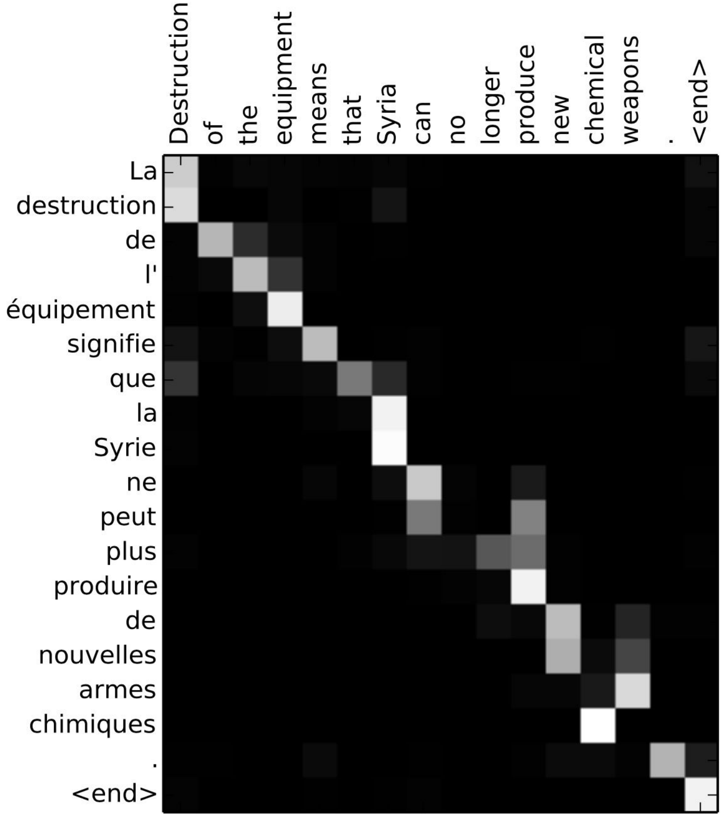
Seq2seq with attention

- Each decoder state
  - Scores prev. hidden states (dot product)
  - Turn into probabilities (softmax)
  - Sum to make the **context vec.**
  - Concatenate with hidden decoder state
  - Output and fed to next step



Why seq2seq with attention

- ❖ Enables one different context for each decoding step
  - ■ No fix-sized bottleneck
- ❖ Provides shortcuts (better gradient flows)
- ❖ More fine-grained -> better interpretability



A typical RNN model today...

"An encoder-decoder bidirectional, multilayered LSTM-based RNN with an attention mechanism"

References

[40] Andrej Karpathy, The Unreasonable Effectiveness of Recurrent Neural Networks, May 21, 2015. http://karpathy.github.io/2015/05/21/rnn-effectiveness/
[41] https://cs.stanford.edu/people/karpathy/deepimagesent/
[42] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780.
[43] http://colah.github.io/posts/2015-08-Understanding-LSTMs/
[44] Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).
[45] Jozefowicz, R., Zaremba, W., Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In Proceedings of the 32nd International Conference on Machine Learning(ICML-15) (pp. 2342-2350).

References

[46] Chung, J., Gulcehre, C., Cho, K., Bengio, Y. (2014).Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.
[47] https://calvinfeng.gitbook.io/machine-learning-notebook/supervised-learning/recurrent-neural-network/recurrent_neural_networks/
[48] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." arXiv preprint arXiv:1409.3215 (2014).
[49] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).
[50] Britz, Denny, Anna Goldie, Minh-Thang Luong, and Quoc Le. "Massive exploration of neural machine translation architectures." arXiv preprint arXiv:1703.03906 (2017).

References

[51] Mikolov, Tomas, et al. Distributed Representations of Words and Phrases and their Compositionality.
[52] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

Dario Garcia-Gasulla (BSC)
dario.garcia@bsc.es