Introduction to Large Language Models

Albert Gatt & Ayoub Bagheri

Utrecht University

Goals of this short course

- Main goal: understanding the fundamentals of (large) language models

→The language modeling objective

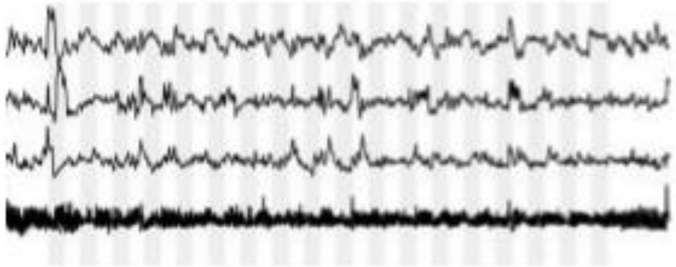
→Sequences and the encoder-decoder architecture

→Attention

→Transformer models

Language modelling

It's about sequences

- "This morning I took the dog for a walk." sentence
-  medical signals
-  speech waveform

Sequence modelling

This morning I took the dog for a……. walk

This morning I took the dog to the……vet

Given: history/context

predict what comes next

Approach 1: fixed window (n-grams)

- Model the sequence in terms of smaller sub-sequences of length *n* (e.g. n=1, 2 or 3)

- Unigram (1-gram): $$p\left(x_1, \ldots., x_n\right) \approx \prod_{i=1}^{n} p(x_i)$$

- Bigram: $$p\left(x_n | x_1, \ldots., x_{n-1}\right) \approx p(x_n | x_{n-1})$$

Would a bigram model handle this?

$$p\left(x_n|x_1, \ldots, x_{n-1}\right) \approx p(x_n|x_{n-1})$$

This morning I took the dog for a……. walk

This morning I took the dog to the……vet

Given: history/context

p(walk | this morning…) = p(walk | a)
p(vet | this morning…) = p(vet | the)

But surely, predicting *walk/vet* also depends on *dog*?

With a simple Markov model like this, we cannot model long-term dependencies.

Long-distance dependencies

Word probabilities:

In Spain, I ate a lot of paella and learnt some Spanish.

Pronouns and their antecedents:

John told me he was leaving on the 24th.

Mary told me she was leaving on the 24th.

Bag of words?

- Idea: treat the whole sequence as a multiset (“bag”), where each element is represented by a count.
  - We can also weight such counts in various ways, e.g. TF/IDF

This morning I took the dog for a

- But BoW doesn’t preserve order. These two are equivalent:
  - *In Spain, I ate a lot of paella and learnt some Spanish.*
  - *I ate some paella and learnt a lot of Spanish in Spain.*

Suppose we use a really big fixed-window?

- More or less, like using n-grams with larger values for *n*
  - E.g. consider previous 5 words

This morning I took the dog to the……vet

Long-distance dependencies (again)

Word probabilities:
In Spain, I ate a lot of paella and learnt some Spanish.

Pronouns:
John told me he was leaving on the 24th.
Mary told me she was leaving on the 24th.

Distance is not fixed:
John, who is my mother's brother, told me he was leaving on the 24th.
Mary, who is my father's sister, told me she was leaving on the 24th.

Summary: The challenges of sequences

Ideally, we want to deal with:

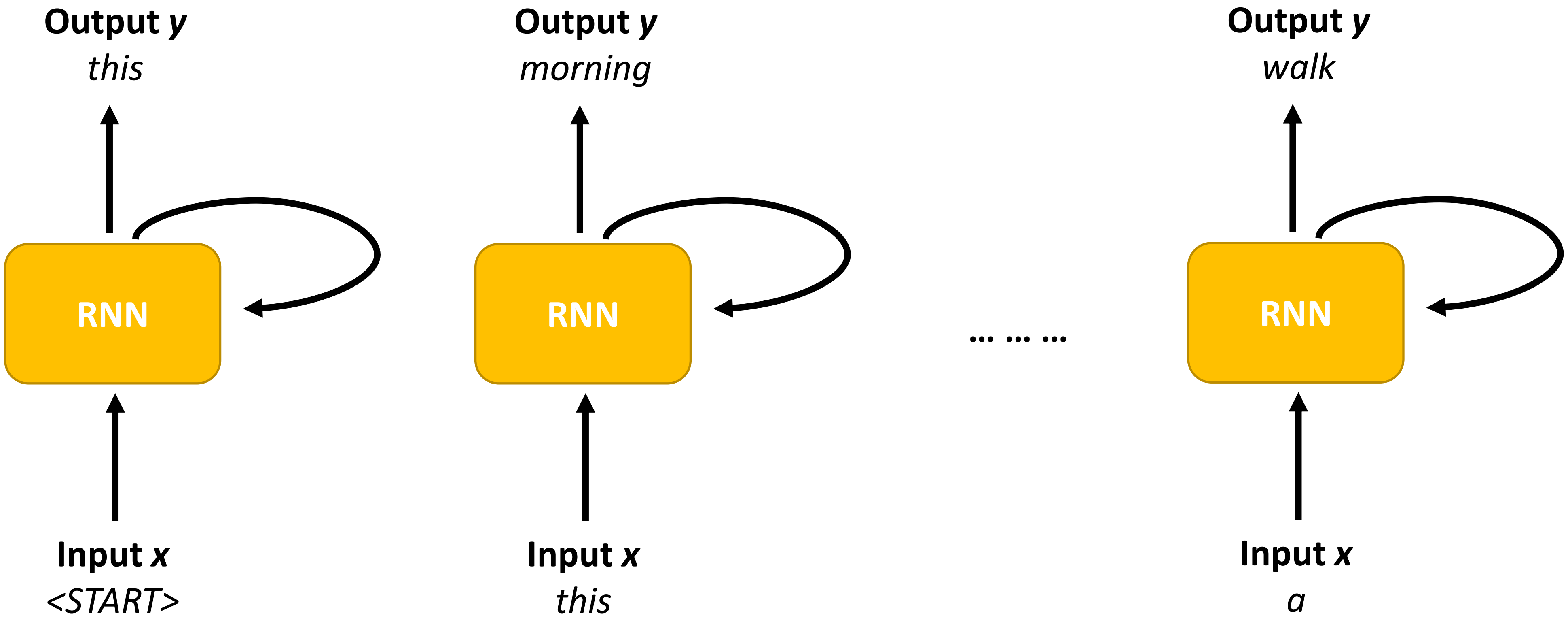
- variable length
- order preservation
- long-distance dependencies
- parameter sharing across the sequence

Recurrent networks

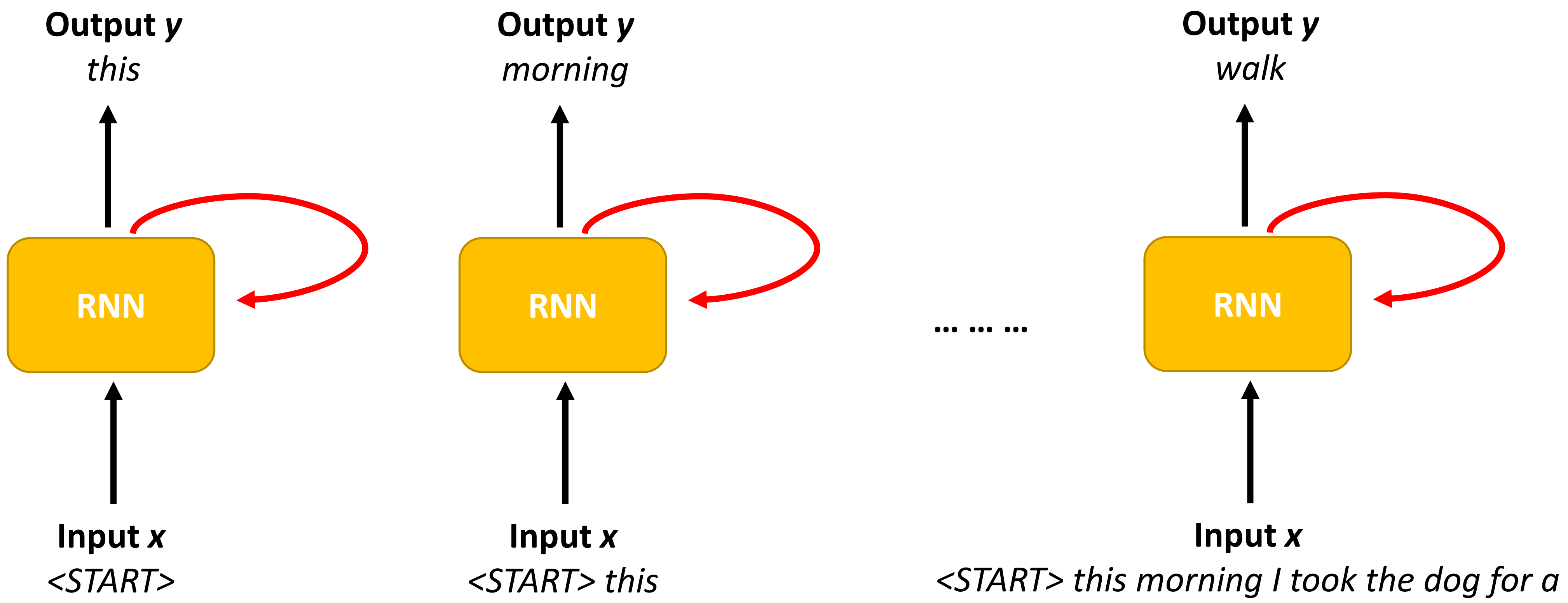
Recurrent Neural Networks (RNNs)

- Rationale:
  - Rather than fixing the amount of history our network can handle, we allow it to **accumulate a representation over time**.
  - This can work over arbitrary sequences.
  - Hopefully, it will also encode similar things in similar ways (less representational redundancy).
  - In the accumulated representation, it should also capture dependencies between elements at different (possibly distant) time-steps.

Recurrent Neural Network

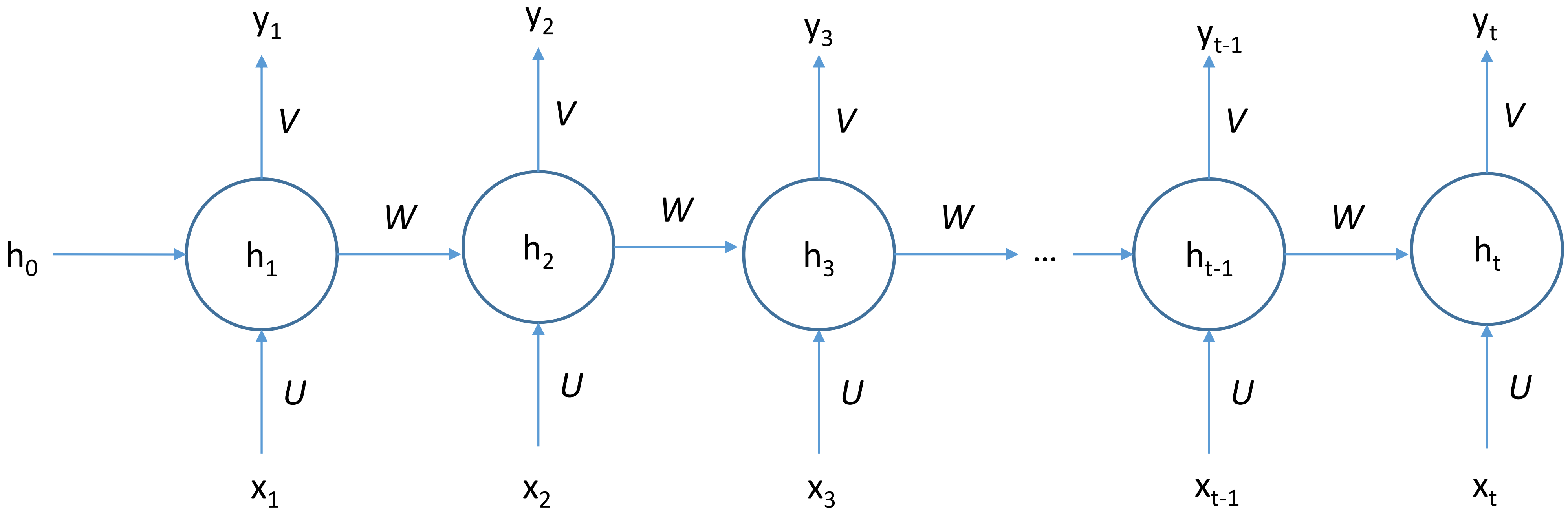


Recurrent Neural Network



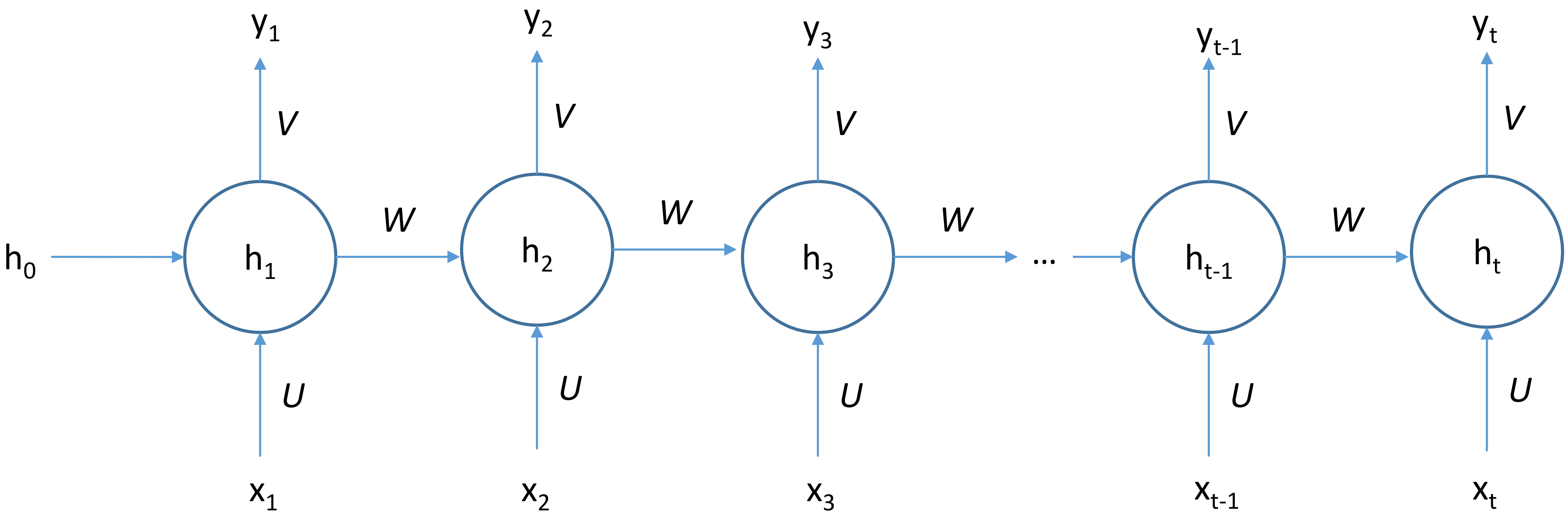
RNN maintains a hidden state across timesteps which accumulates the sequence representation.

Unrolled RNN



- Processes a sequence $x_1, \ldots, x_t$ via hidden units $h_1, \ldots, h_t$ to yield an output sequence.
  - Key property: **share parameter matrices U, W and V across time.**

Recurrent neural network: parameters



- **Recurrence formula**

Often, g = tanh

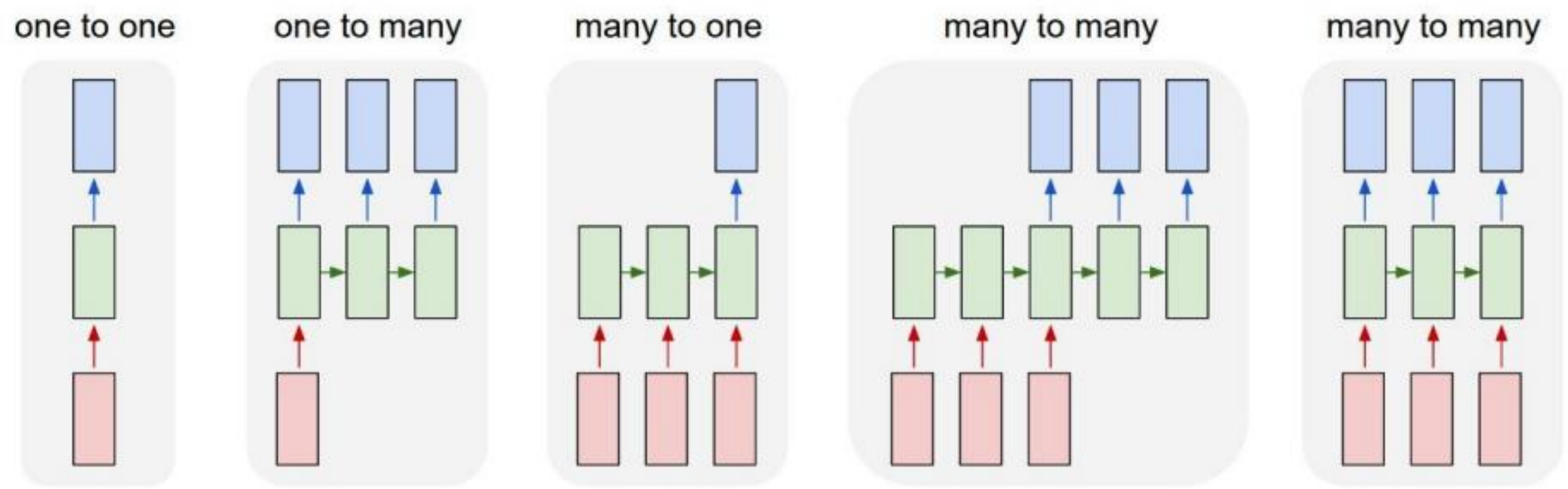
$$h_t = f(h_{t-1}, x_t) = g(Wh_{t-1} + Ux_t)$$

$$y_t = Vh_t$$

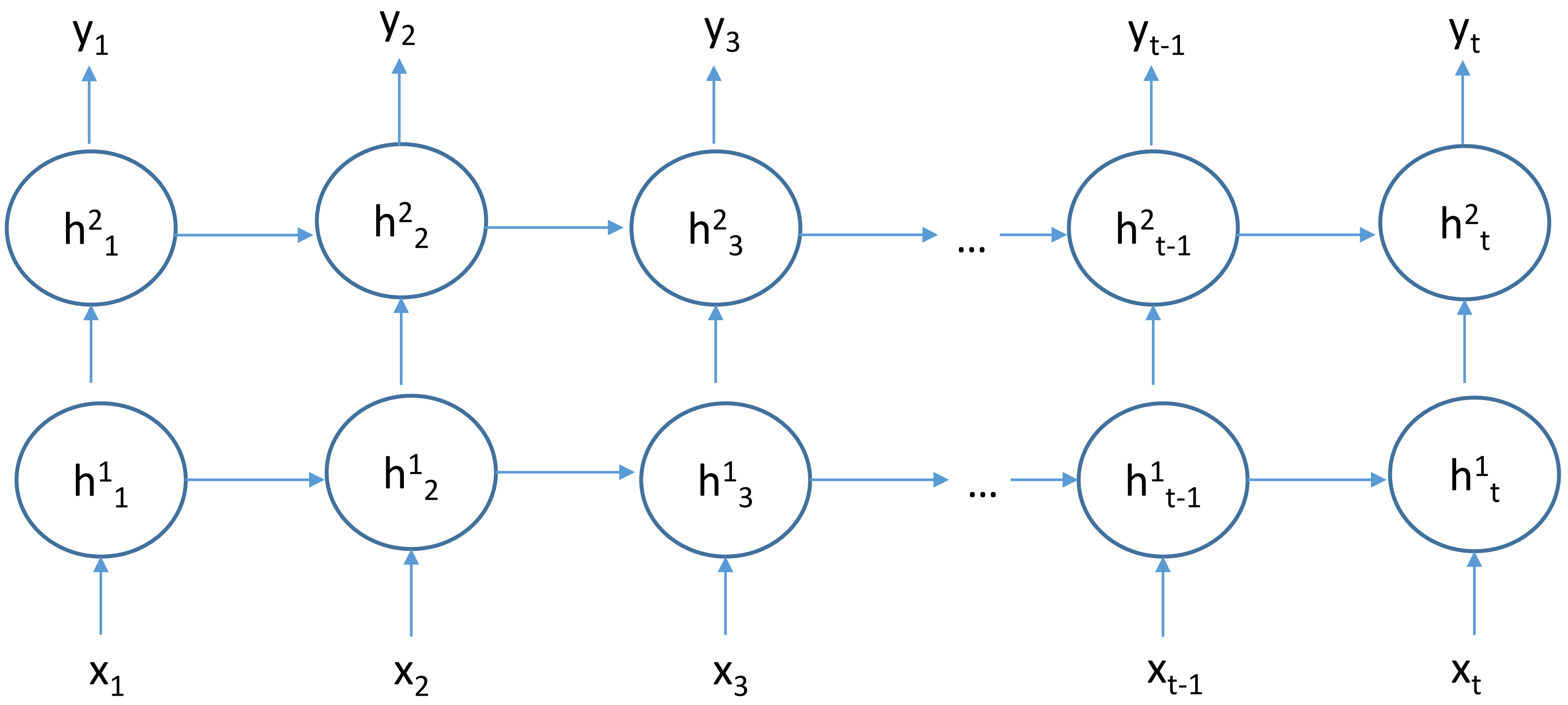
Previous hidden state
Shared W

Input vector at current timestep.
Shared U

Recurrent Neural Networks: Process Sequences



Depth



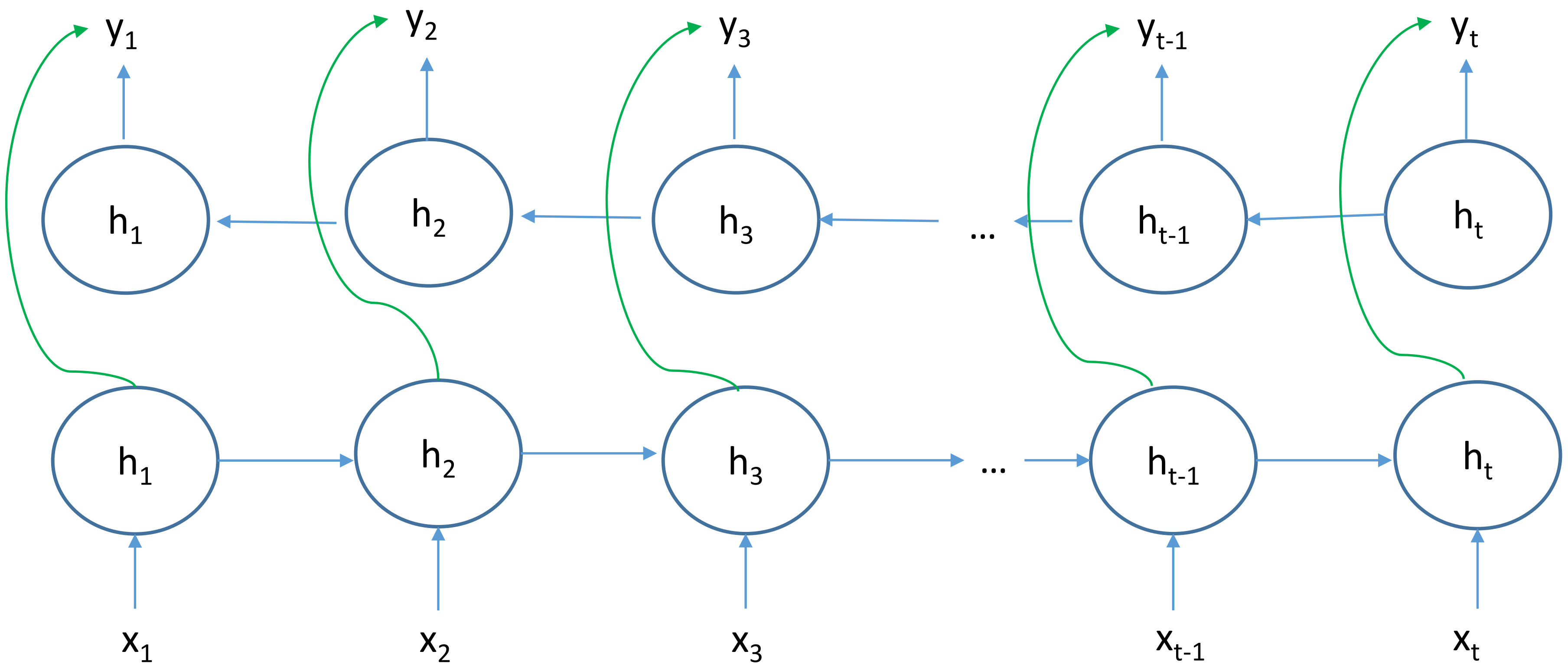
RNN with two layers.

- We can stack multiple hidden layers in the RNN and connect them.
- Each layer has its own weight matrix for hidden states (W) and inputs (U) are:

$$h_t^l = g\left(W^l h_{t-1}^l + U^l h_t^{l-1}\right)$$

Input from the previous layer (= x at layer 1)

Bidirectionality



$$
\begin{aligned}
h_t &= g\left(W h_{t-1} + U_t^x\right) \\
\overline{h}_t &= g\left(\overline{W} h_{t+1} + U_t^x\right) \\
y_t &= V\left[h_t; \overline{h}_t\right]^T
\end{aligned}
$$

Gating (brief outline)

RNNs can handle arbitrary sequences, but they do forget!

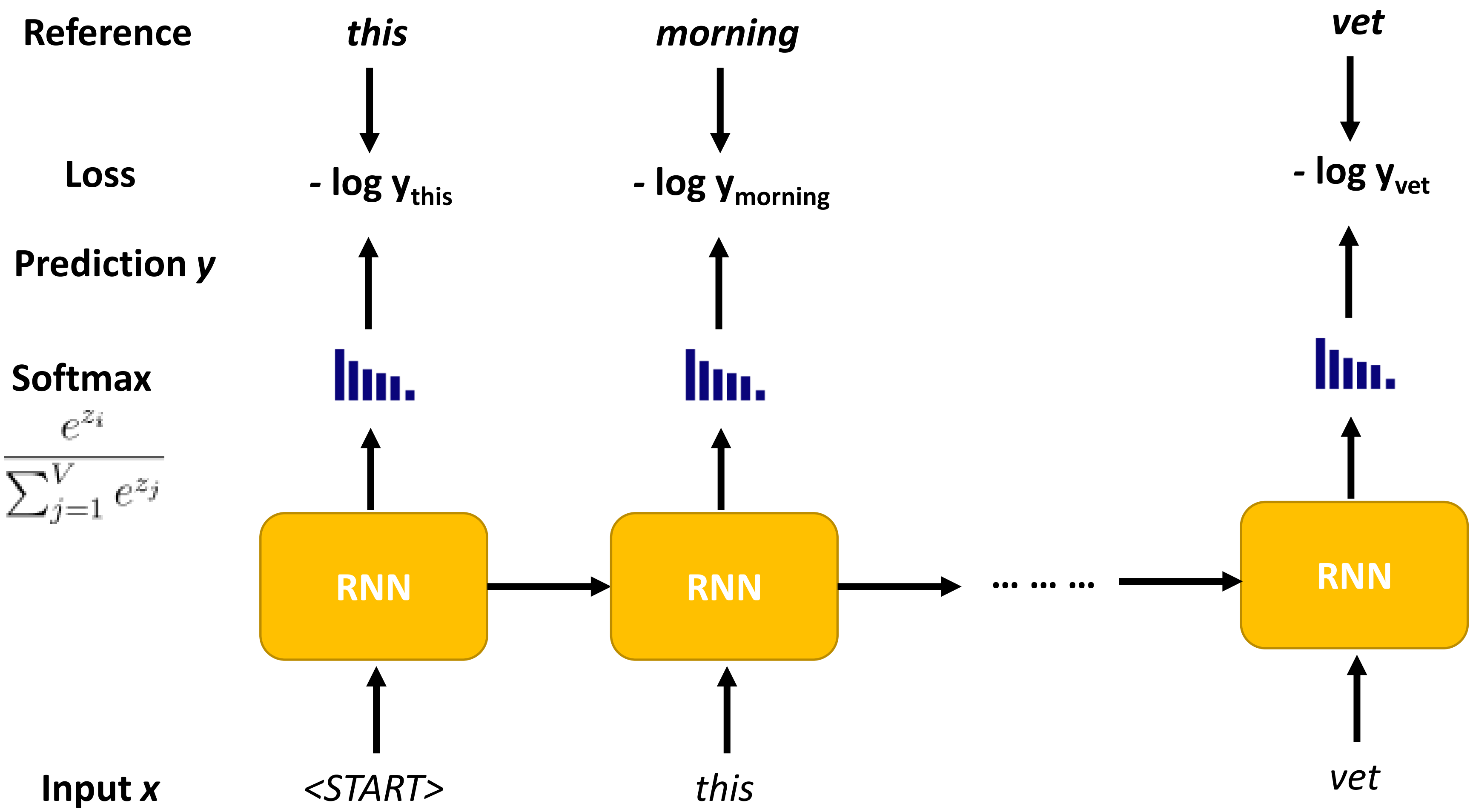
- During training, long-distance dependencies are “lost” as the impact of early elements is diminished over time.

Gated Recurrent Unit (GRU)

Long Short-term Memory (LSTM) network

- Two variations of RNNs use “gates” to allow the network to selectively retain or “forget” information.

Training with cross-entropy loss

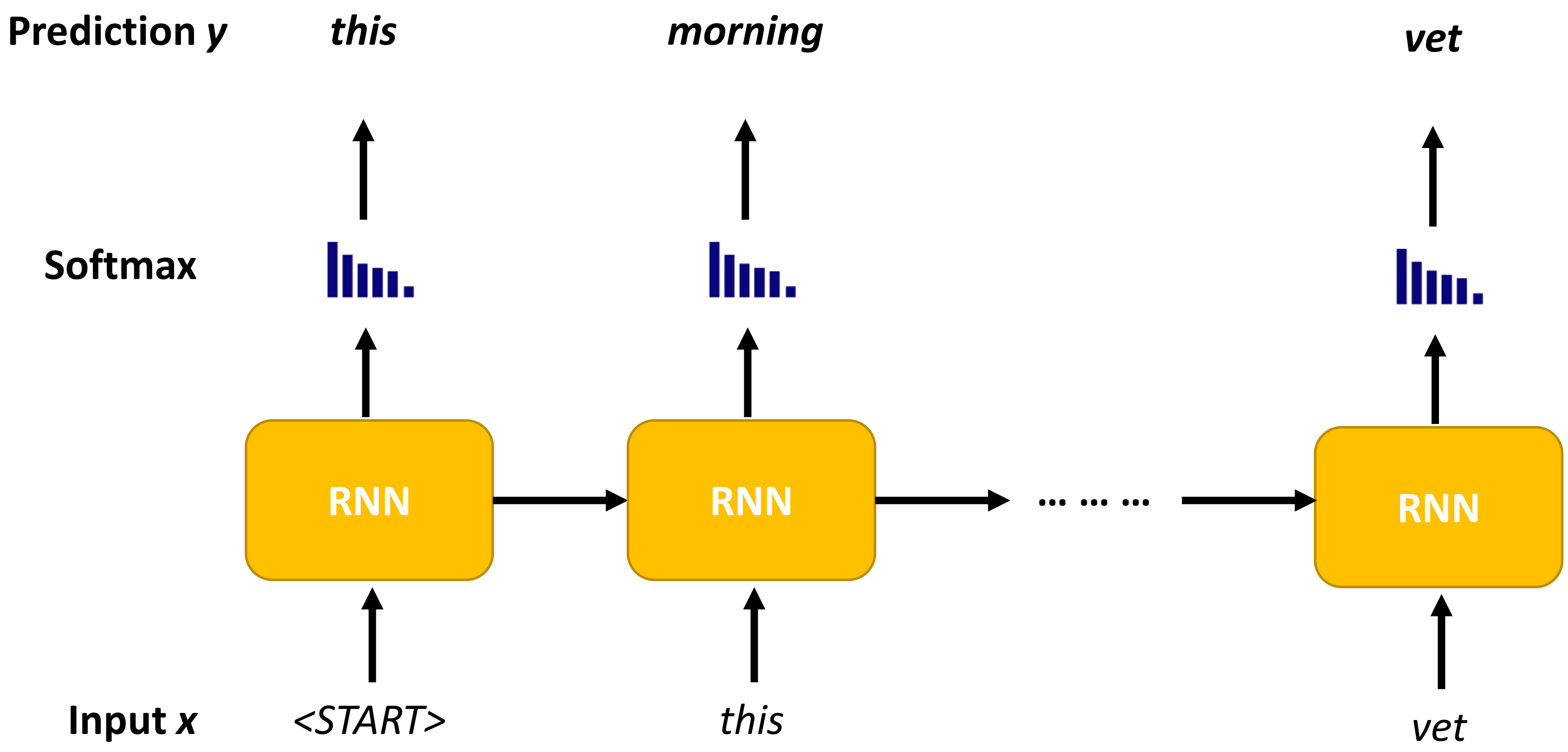


Loss is averaged over the sequence:

$$\frac{1}{T}\sum_{t=1}^{T} L_{CE}$$

$$L_{CE} = -\log \hat{y}_t$$

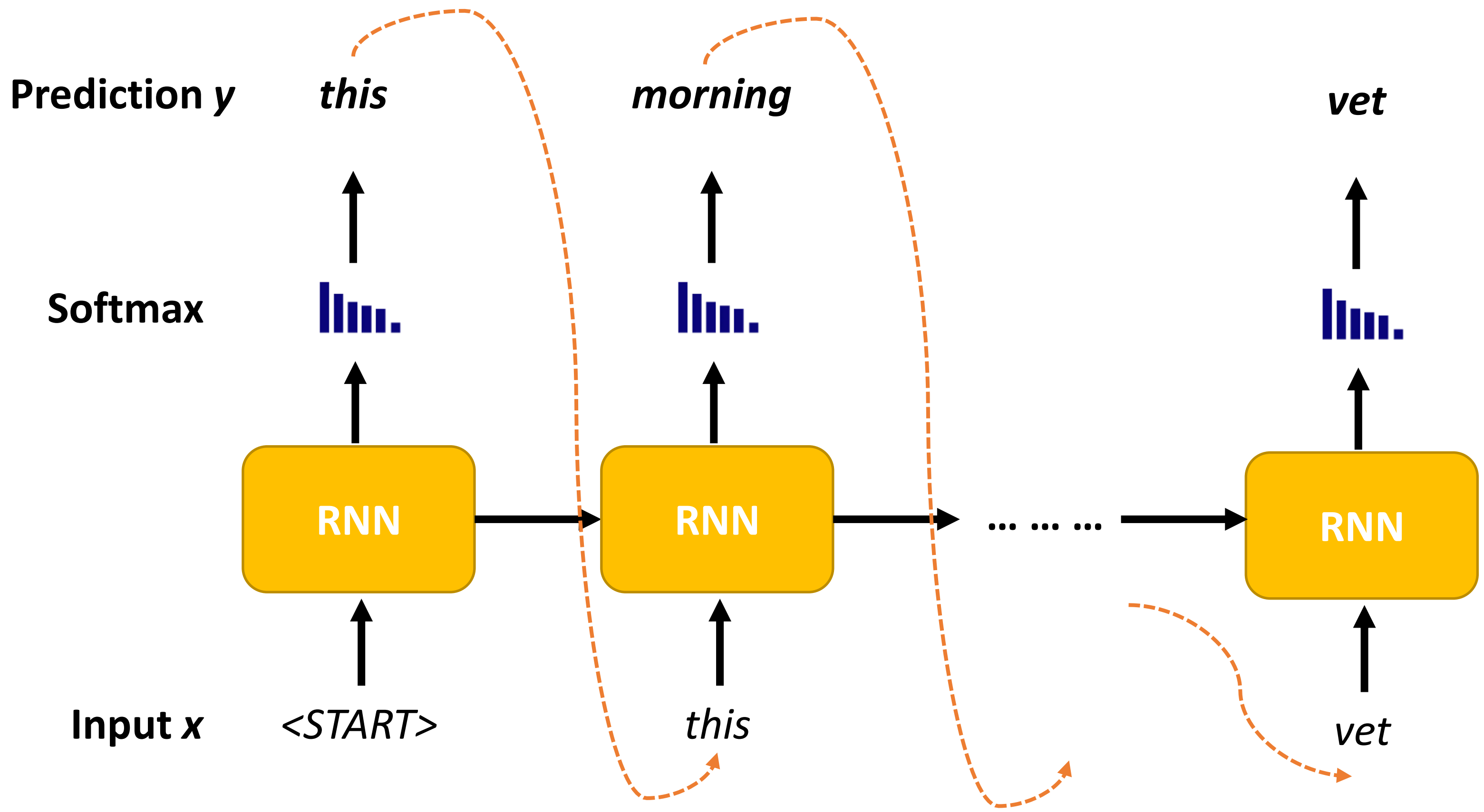
Prediction with a trained RNN



Encoder-Decoder Architecture

Causal ("auto-regressive") generation

- Basic idea: condition next word prediction on the preceding word, plus the hidden state.



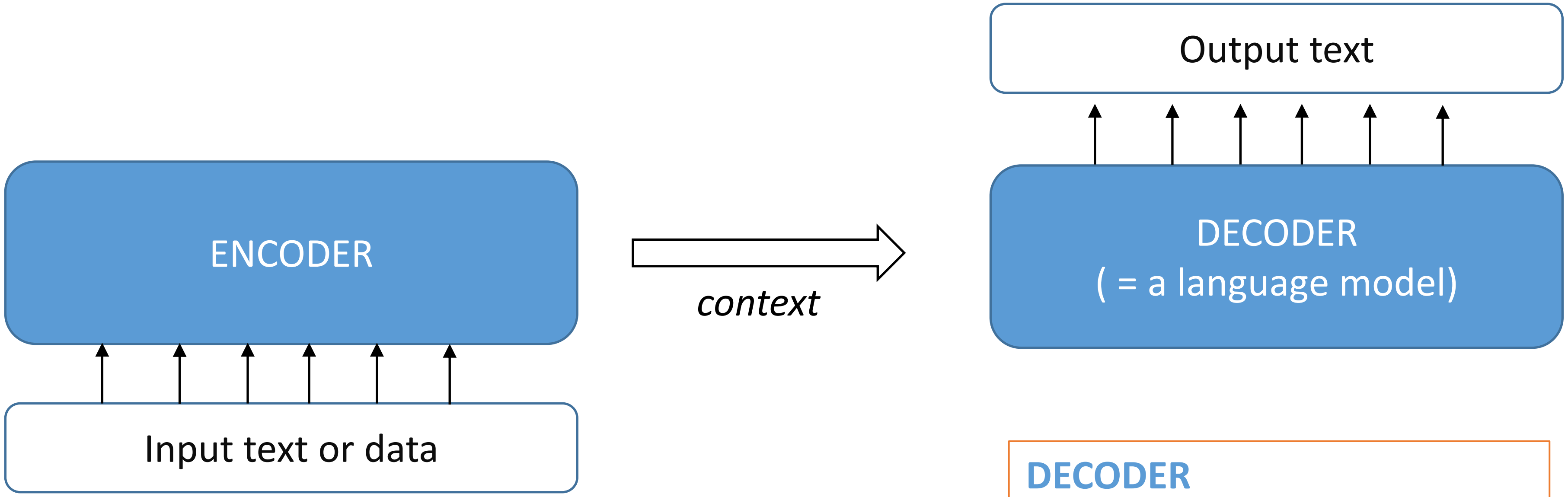
Causal ("auto-regressive") generation

- At each time step *t*, sample from the vocabulary V, and choose the most likely next element, based on:
    - Current hidden state (which accumulates the representation up to *t-1*)
    - Previous word generated at *t-1*

- Notice, however, that this generates random sequences.
    - What is the text being generated actually about?

- Suppose we want to generate from some input?
    - Idea: condition word choice based on the input, as well as previously generated words.

Encoder-Decoder architecture



ENCODER

- Transform input to a vector representation (the **context**).
- Representation is available in the network's hidden layer.

DECODER

- Conditioned on hidden state of the encoder.
- Produce output by sampling from a probability distribution.
- At each time-step, we predict the next token.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems 27 (NIPS'14)*, 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. https://doi.org/10.3115/v1/D14-1179

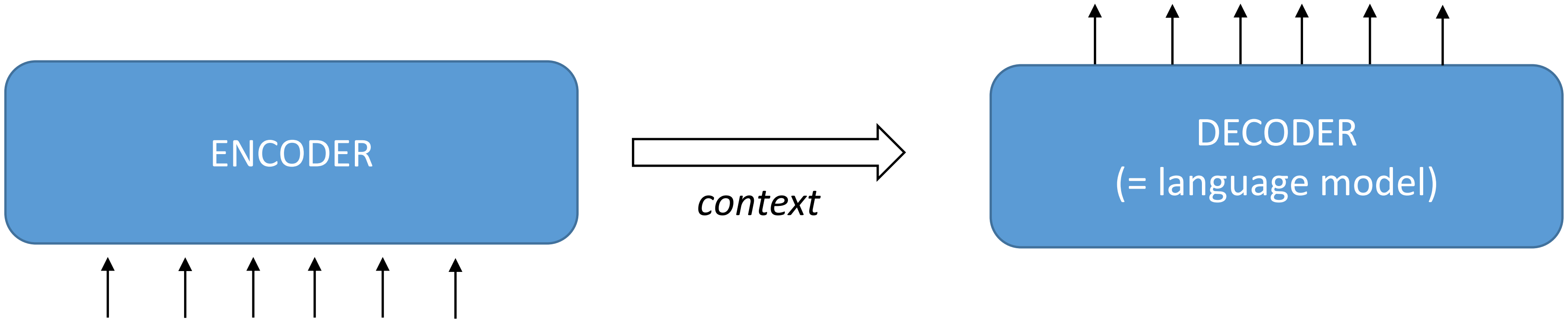
Encoder-Decoder architecture

(b) English text

Trane, which was founded on January 1st 1913 in La Crosse, Wisconsin, is based in Ireland. It has 29,000 employees.

(c) Russian text

Компания "Тране", основанная 1 января 1913 года в Ла-Кроссе в штате Висконсин, находится в Ирландии. В компании работают 29 тысяч человек.




```
<entry category="Company" eid="Id21" shape="(X (X) (X) (X) (X))" shape_type="sibling" size="4">
    <modifiedtripleset>
        <mtriple>Trane | foundingDate | 1913-01-01</mtriple>
        <mtriple>Trane | location | Ireland</mtriple>
        <mtriple>Trane | foundationPlace | La_Crosse,_Wisconsin</mtriple>
        <mtriple>Trane | numberOfEmployees | 29000</mtriple>
    </modifiedtripleset>
</entry>
```


Example from the WebNLG challenge
https://webnlg-challenge.loria.fr/challenge_2020/

Encoder-Decoder architecture

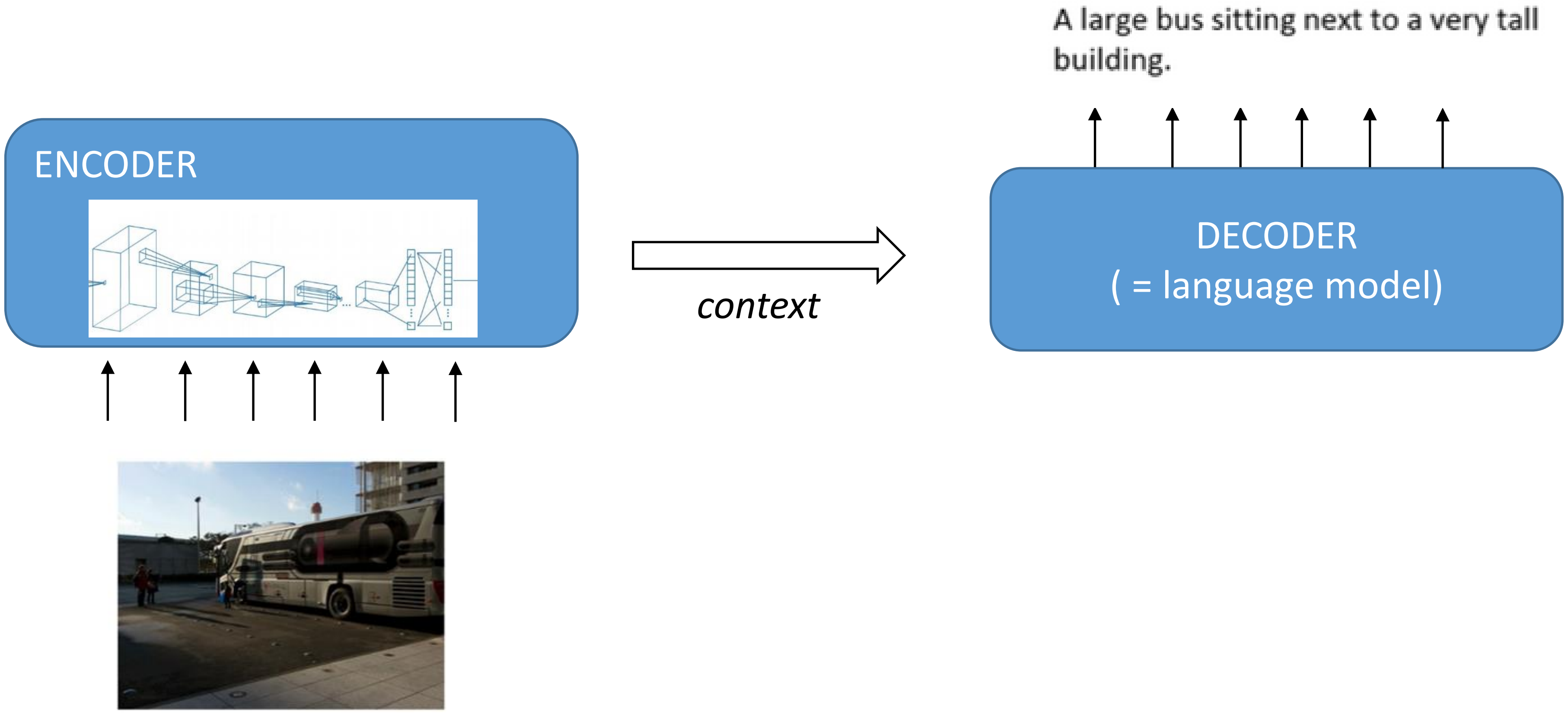


Image and caption taken from the COCO Challenge.
https://cocodataset.org/#captions-2015

Where do we factor in the context?

Various ways to do this. Here are two:

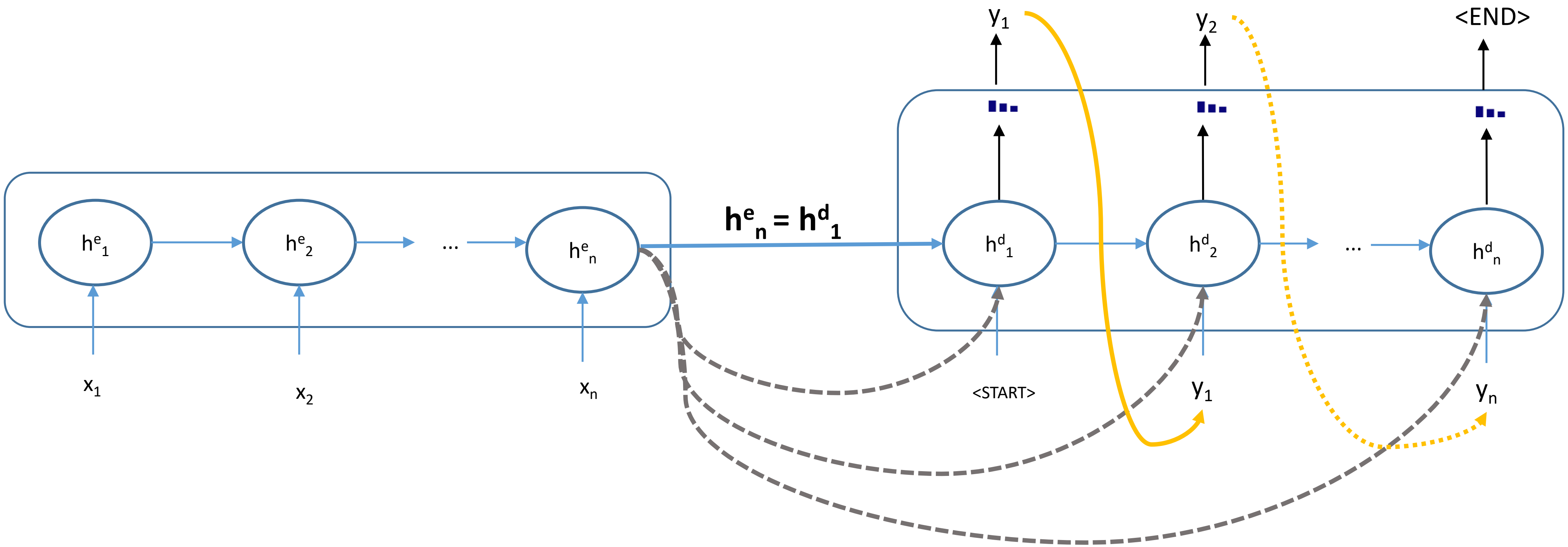
Initialise

- Initialise the decoder with the encoder’s last hidden state
- During decoding, decoder predicts the next element with the previous hidden state and the predictions generated so far.

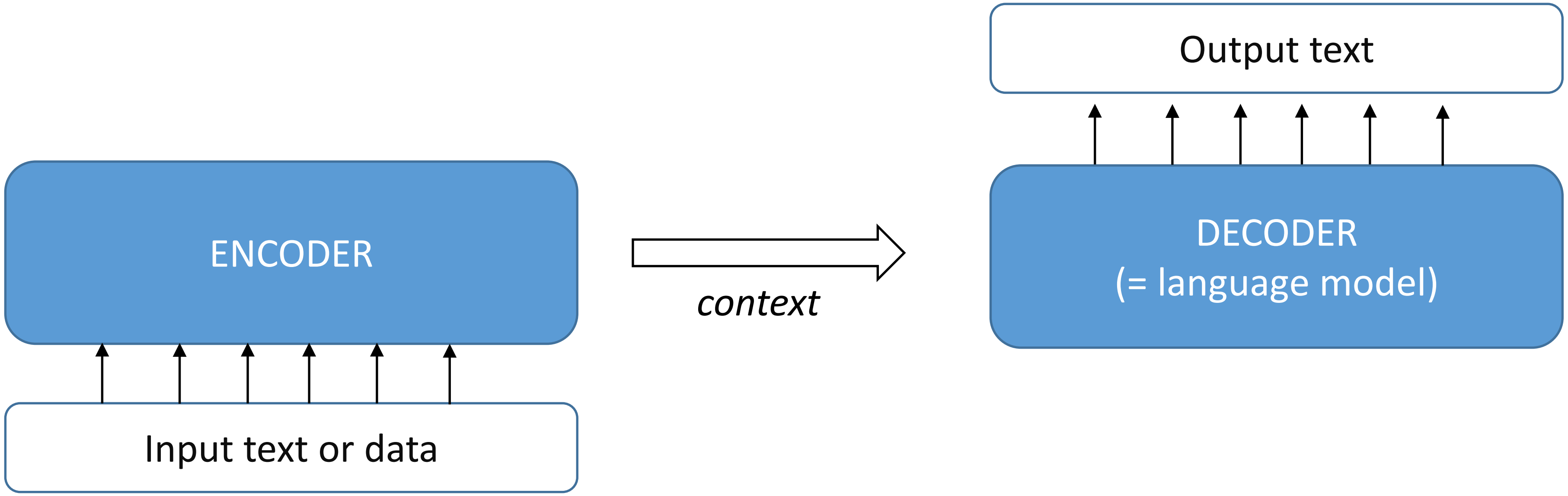
Use context at each step

- Inject the context into each time-step of the decoder
- At each timestep, decoder predicts based on its previous hidden state, predictions generated so far, and the context.

The encoder-decoder model at inference time (NB: context injected at each step)



Where does attention come into the picture?

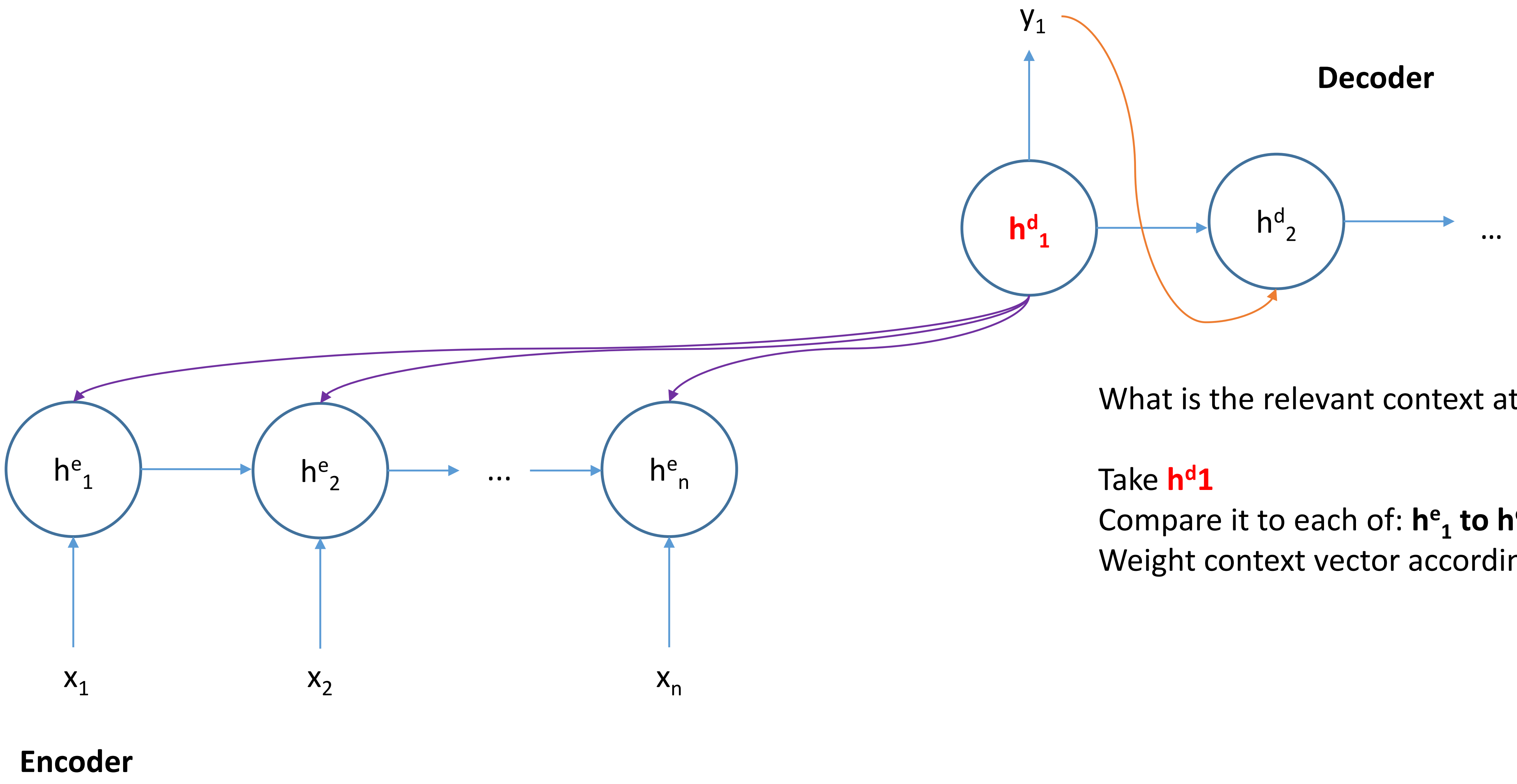


The original E-D creates a **bottleneck**: All the input is compressed into a dense representation, which is used throughout decoding.

Attention mechanisms allow the decoder to learn to differentiate between parts of the input context. So at each time-step, we train it to pay more attention to relevant portions of the input.

Transformers are based on a generalization of the attention mechanism.

Attention in Encoder-Decoder models



What is the relevant context at $\mathbf{h^d_2}$?

Take **h^d1**

Compare it to each of: **h^e_1 to h^e_n**

Weight context vector accordingly

Attention in Encoder-Decoder models

- Suppose input is of length n. Then the Encoder has n states, one for each time-step (word etc): $h^e_1 ... h^e_n$
- Let c_i be the decoder context at timestep i. We want this to reflect **how relevant** each encoder state is to the decoder state h^d_{i-1}.
  - We capture this by comparing h^d_{i-1} to each encoder state h^e_j.

Attention in Encoder-Decoder models

1. Capture their similarity: compare the decoder state to each encoder state:

$$\text{score}(h_i^d, h_j^e) = h_i^d \cdot h_j^e$$

2. Normalise scores using softmax (turn them into a distribution):

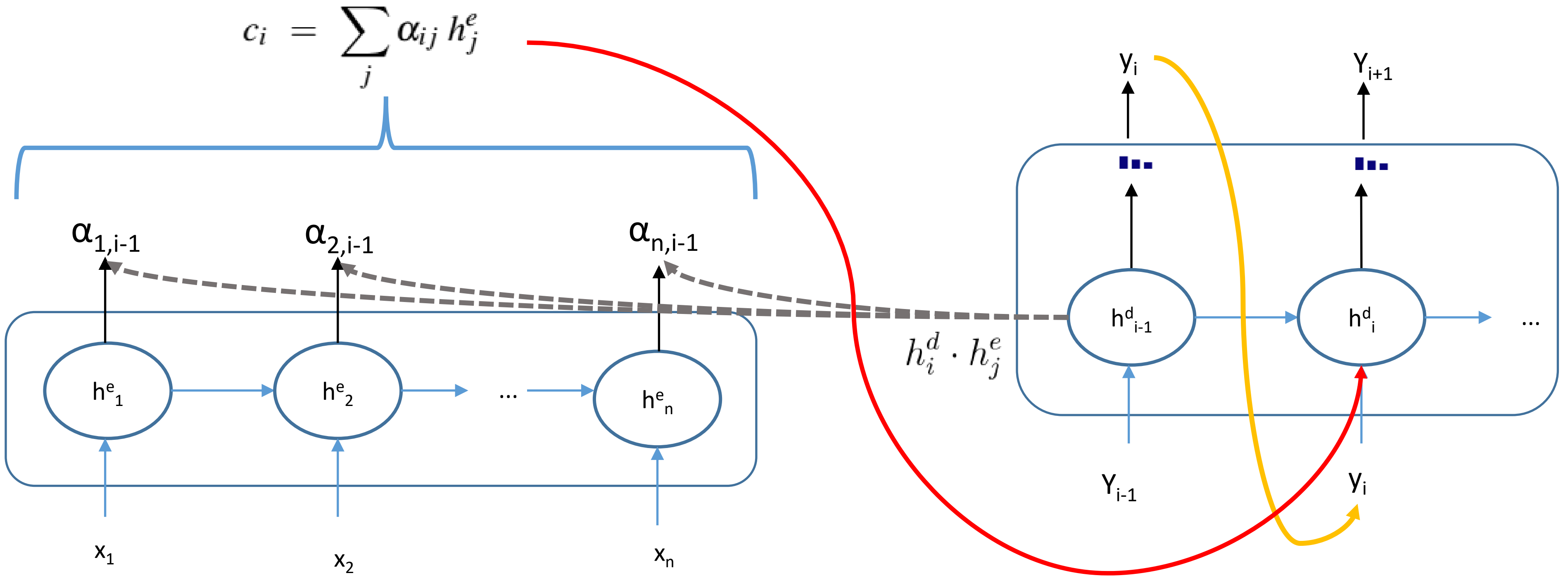
$$\alpha_{ij} = \text{softmax}(\text{score}(h_i^d, h_j^e), \forall j \in e)$$

3. Use that distribution to compute a weighted average over all the hidden encoder states:

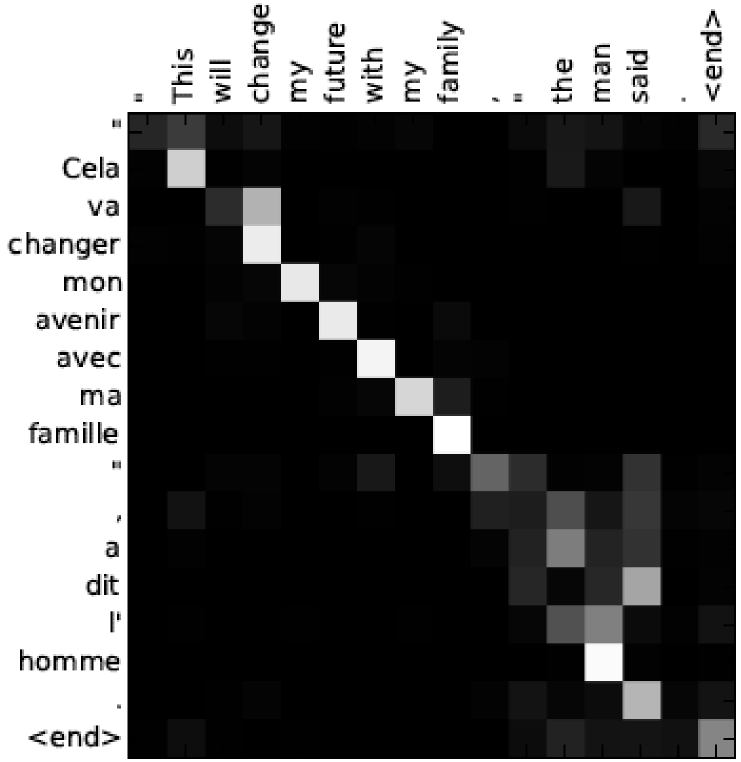
$$c_i = \sum_j \alpha_{ij} \, h_j^e$$

What does this do?

- Hopefully, during training, we achieve a way to identify, at each time-step in the decoder, which part of the source encoding is most relevant.



Attention as "alignment"



A stop sign is on a road with a mountain in the background.

A woman holding a clock in her hand.

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation By Jointly Learning To Align and Translate. *Proceedings of the International Conference on Learning Representations (ICLR'15)*, 1–15. https://doi.org/10.1146/annurev.neuro.26.041002.131047

Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., & Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *Proceedings of the International Conference on Learning Representations (ICLR'15)*. http://arxiv.org/abs/1502.03044

Summary for today

- Language modelling objective
  - Basically, word prediction given a preceding context.
- Recurrent networks
  - Architecture to handle sequences of arbitrary length.
- Endoder-Decoder
  - Architecture (originally with RNNs) to drive a language model to generate, conditioned on input context.
- Attention
  - Key step to condition the LM selectively on different parts of the input, at each timestep.

What’s next?

- The current generation of “large language models” is based on Transformer architectures.
- These generalise the notion of attention, while removing recurrence completely.

- Next time:
  - Generalisation of attention
  - Transformers and self-attention
  - Concrete examples of transformer-based LMs.