

# Natural Language Understanding, Generation, and Machine Translation

## Lecture 8: Sequence-to-sequence Models with Attention

---

Frank Keller

Week of 31 January 2022 (week 3)

School of Informatics

University of Edinburgh

[keller@inf.ed.ac.uk](mailto:keller@inf.ed.ac.uk)

Based on slides by Adam Lopez, Rico Sennrich.

# Overview

Encoding and decoding sequences

RNN for Translation

Translation as String Completion

Encoder-Decoder Architecture

Attention

Attention Distribution over Source Words

Bidirectional RNN

Attention and Alignment

Reading: Sections 7–9 of Neubig (2017).

# Agenda for Today

**Last week:** We saw that we could model a probability distribution over a strings, like  $w = w_1 \dots w_{|w|}$ :

1. Use the chain rule,  $P(w) = \prod_{i=1}^{|w|+1} P(w_i \mid w_1, \dots, w_{i-1})$ , and
2. Use *universal function approximators* in the form of neural networks to model the conditional distributions

$$P(w_i \mid w_1, \dots, w_{i-1})$$

The second idea brings many benefits over classic  $n$ -gram models. It gives us parameter sharing and representation learning, and it frees us from zero probabilities and independence assumptions.

**Today:** We will see how to model *conditional probability distributions* like  $P(y \mid x)$ , where  $x$  and  $y$  are both sequences of discrete symbols, aka strings.

# Encoding and decoding sequences

---

## How would you model translation with $n$ -grams?

**Input:** Så varför minskar inte vi våra utsläpp?

**Output:** So why are we not reducing our emissions?

Let  $x$  be the *source* (Swedish) sentence,  $y$  be the *target* (English) sentence.

$$x = x_1 \dots x_{|x|}$$

$$y = y_1 \dots y_{|y|}$$

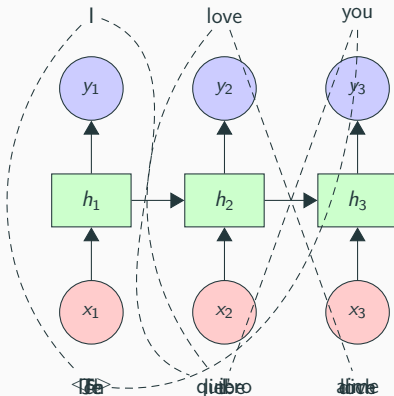
How can we define  $P(y \mid x)$ ?

Expand using the chain rule:

$$P(y \mid x) = \prod_{i=1}^{|y|+1} P(y_i \mid y_1, \dots, y_{i-1}, x_1, \dots, x_{|x|})$$

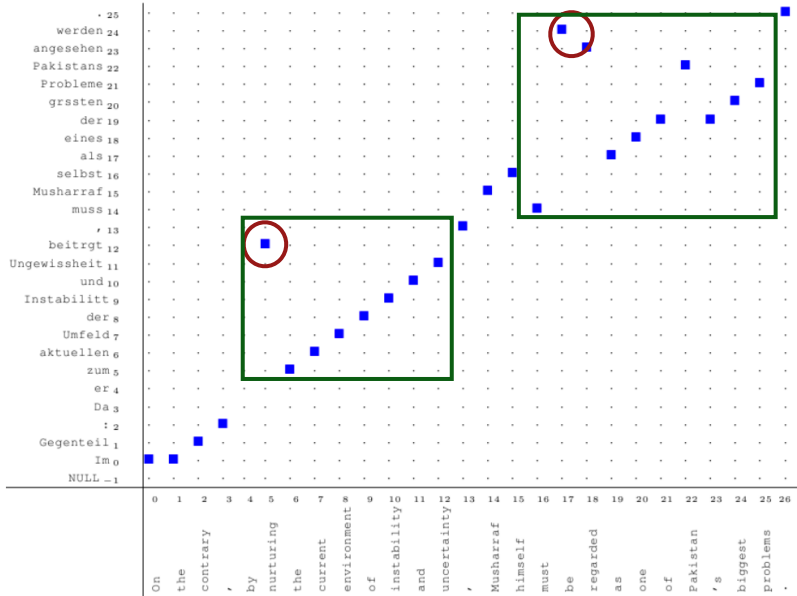
# Idea: use RNN language models for translation

shift the word by 1

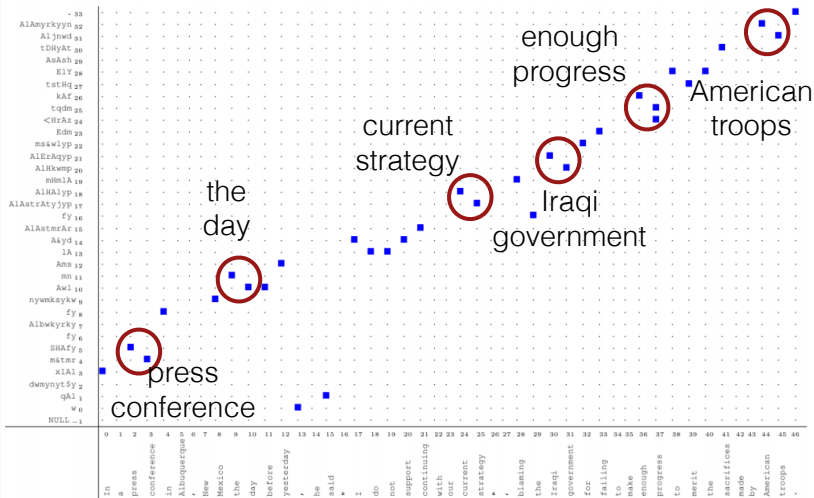


Word order is different! Different number of words!

# Syntax is very different across languages

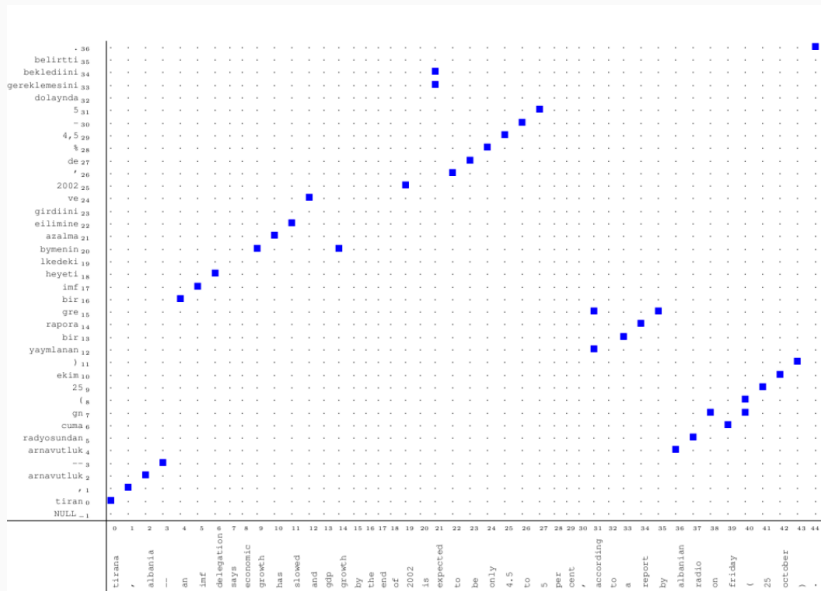


# Syntax is very different across languages





# Syntax is very different across languages

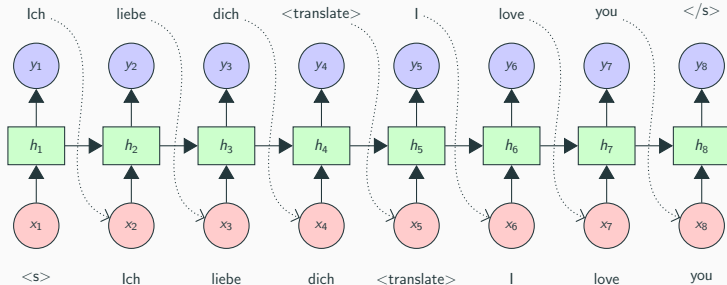


# Syntax is very different across languages

	SUBJECT	VERB	OBJECT
English	I	♥	NY
Welsh	♥	I	NY
Malagasy	♥	NY	I
Japanese	I	NY	♥
Hixkaryana	NY	I	♥
Nadëb	NY	♥	I

Variation in word order is a large area of research in linguistics. In the 1960's, a linguist from Brooklyn, Joseph

# Idea: model translation as string completion



Does this design make sense? Why or why not?

**mix up the two language**

# Translation and Language Modeling are different

Language model:

$$p(y) = \prod_{i=1}^n p(y_i | y_1, \dots, y_{i-1})$$

Translation model:

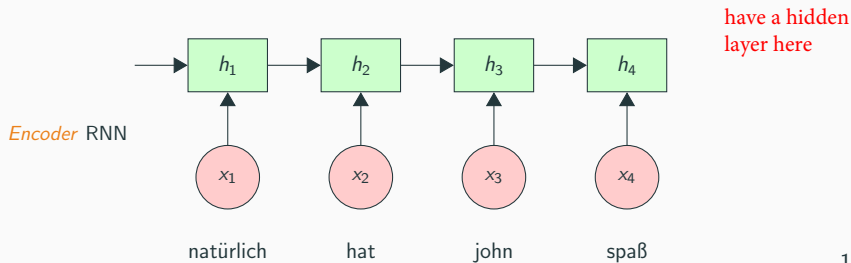
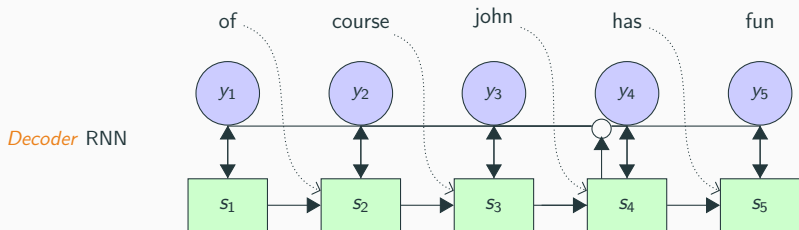
$$p(y | x) = \prod_{i=1}^n p(y_i | y_1, \dots, y_{i-1}, x_1, \dots, x_{|x|})$$

We could treat sentence pair as one long sequence, but:

- This is really a model of  $P(x, y) = P(x)P(y | x)$ .
- But we do not care about  $P(x)$ !
- The source and target language may have very different vocabulary and morphosyntactic structure.

**Insight.** We need two RNNs, one for source and one for target.

# Use one RNN to encode source, another to decode target



# Formalizing the Encoder-Decoder approach

An RNN is a function of a hidden state and new input vector, returning a hidden state (so any RNN variant can be used).

Encoder:

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

the first time step of  
decoder initialized by the  
last time step of encoder

Decoder:

$$\mathbf{s}_i = \text{RNN}_{\text{dec}}(\mathbf{y}_i, \mathbf{s}_{i-1})$$

$$P(y_i \mid y_1, \dots, y_{i-1}, x_1, \dots, x_{|x|}) = \text{softmax}(\mathbf{W} \text{concat}(\mathbf{s}_i, \mathbf{h}_{|x|}) + \mathbf{b})$$

Final encoder hidden state  $\mathbf{h}_{|x|}$  is *context* for decoder softmax.

## Interpreting the Encoder-Decoder approach

- Given a conditional probability  $P(y \mid x)$ , we can design two neural architectures: one to *encode*  $x$  into a hidden representation, and one to *decode*  $y$  from that representation.
- In machine translation, RNNs are a reasonable choice for both encoder and decoder.
- The hidden representation from the encoder is supplied as input to the decoder *at every time step* (because vectors).
- But idea is general. Design encoder and decoder to suit task.
- The encoder RNN does not need to compute  $P(x)$ . It need only compute a representation.

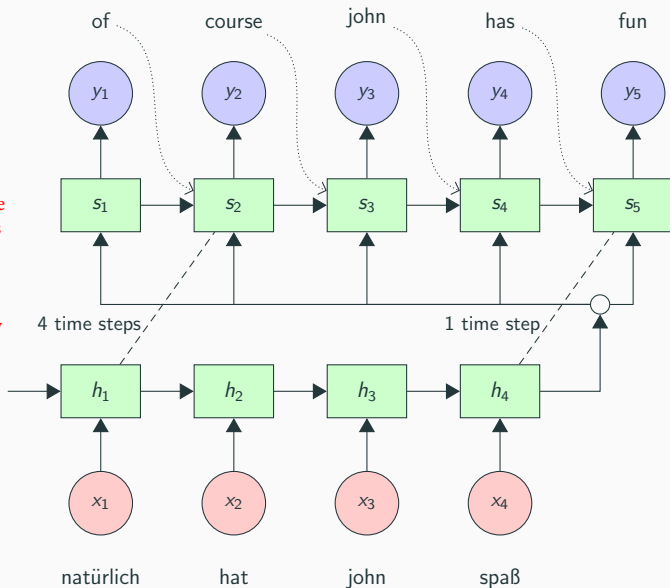
# Q: Does this architecture make sense for translation?

Decoder RNN

bidirectional model, give more weight to previous states.

Learn the weight (attention) to know how important this hidden state contributes to the prediction

Encoder RNN



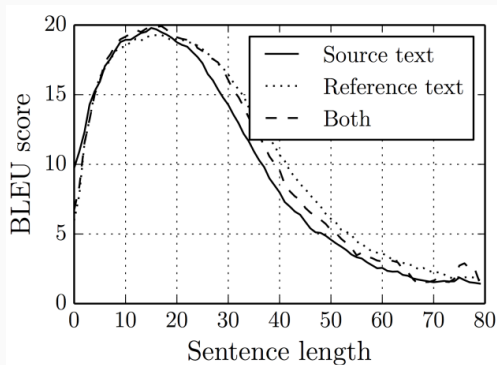


## Encoder-decoder models must consider structure of input

- Hidden representation is a *bottleneck*: it must be able to represent a sentence of any length, but its size is fixed.
- RNNs *and their variants* suffer from vanishing gradients.
- So, representation from the encoder also has a *recency bias*: it mostly represents final words of the input sentence.
- Target words whose corresponding source words are at the beginning of the sentence therefore depend on information that originates further away in the computation graph.

# Encoder-decoder models must consider structure of input

- Empirically, this means the model *forgets* first words of input!
- Reversing input order makes model forget final words of input.



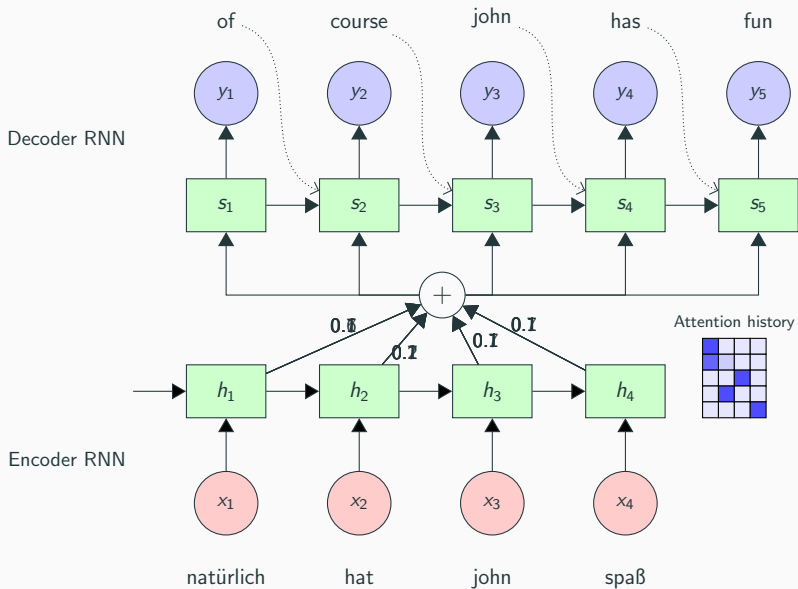
Source: <https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/>

Q. What can we do?

# Attention

---

# Decoder attends to relevant input at each step



## Attention computes a distribution over source words

In decoder softmax, the *context* is now a *weighted average* of source hidden state vectors:

$$P(y_i \mid y_1, \dots, y_{i-1}, x_1, \dots, x_{|x|}) = \text{softmax}(\mathbf{W} \text{concat}(\mathbf{s}_i, \mathbf{c}_i) + \mathbf{b})$$

different from just using the last state (i.e.  $\mathbf{h}_4$ )

$$\mathbf{c}_i = \sum_{j=1}^{|x|} \alpha_{i,j} \mathbf{h}_j$$

Notice that  $\alpha_i$  is a *distribution* over elements of  $x$ . But the length of  $x$  can vary! So how can we compute it? A simple solution:

more similarity between  $\mathbf{s}_i$  and  $\mathbf{h}_i$ , more attention

$$a_{i,j} = \mathbf{s}_i \cdot \mathbf{h}_j$$

$$\alpha_i = \text{softmax}(\mathbf{a}_i)$$

# Interpreting attention

$$a_{i,j} = \mathbf{s}_i \cdot \mathbf{h}_j$$

$$\alpha_i = \text{softmax}(\mathbf{a}_i)$$

- *Attention* solves a problem: we have a variable length input, and we want to summarize it into a fixed vector, focusing only on the *relevant* parts of the input.
- But relevance depends on current prediction task! So we compute a function of the *output hidden state* (the *query*) and each of the available input vectors (the *keys*).
- This computation can take many alternative forms:

$$a_{i,j} = \mathbf{s}_i \mathbf{V} \mathbf{h}_j \quad \text{(bilinear model)}$$

$$a_{i,j} = \text{FFN}(\mathbf{s}_i, \mathbf{h}_j) \quad \text{(feedforward neural network)}$$

## Aside: RNN hidden states encode words in context

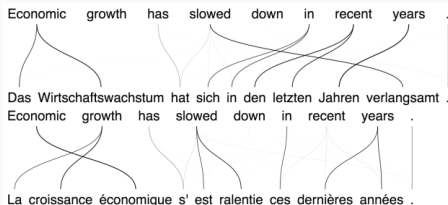
- Attention treats each hidden state of an RNN as a representation of the corresponding source word.
- Last week, we introduced RNNs as a way of summarizing the complete history of a sequence. So  $\mathbf{h}_i$  represents  $x_1 \dots x_i$ .
- Truth is in between: recency bias means that  $\mathbf{h}_i$  mostly represents  $x_i$ , but in the context of  $x_1 \dots x_{i-1}$ .
- This makes sense to disambiguate different meanings of  $x_i$ , but what about context  $x_{i+1} \dots x_{|x|}$ ?

**Idea.** Use *two* encoder RNNs: one left-to-right, one right-to-left. Concatenate their states at position  $i$ . This is a *bidirectional RNN*. Widely used for encoding input strings.

**Q.** Can decoders be bidirectional? **A.** No! Right context missing.

# Attention is not alignment (but alignment inspired attention)

- We are directly modeling  $P(y \mid x)$ .
- We modeled *alignment* as a *latent variable* to work around Markov assumptions:  $P(y, z \mid x)$
- Our neural model has no Markov assumptions. *Attention* is *deterministic*, not *stochastic*. It does not represent a probabilistic choice and we do not marginalize it out.
- Practically: attention looks somewhat like alignment, but since information can flow along recurrent connections, it is more difficult to interpret.





# Attention is a widespread design pattern



a little girl sitting on a bench holding an umbrella.



a herd of sheep grazing on a lush green hillside.



a close up of a fire hydrant on a sidewalk.



a yellow plate topped with meat and broccoli.



a zebra standing next to a zebra in a dirt field.



a stainless steel oven in a kitchen with wood cabinets.



two birds sitting on top of a tree branch.



an elephant standing next to rock wall.



a man riding a bike down a road next to a body of water.

## Encoder-decoder with attention is a common design pattern

We surveyed many applications in the first lecture.

Task	Input type	Output type
Question answering	string	string
Sentiment analysis	string	label
Syntactic parsing	string	tree
Semantic parsing	string	graph (logical form)
Generation	table	string
Image captioning	image	string
Machine translation	string	string

**Q.** How might you use encoder-decoder with attention to model text classification?

# Summary

- To model a conditional probability distribution  $P(y \mid x)$ , use a neural architecture to *encode*  $x$ , and another to *decode*  $y$ .
- A *bidirectional* encoder uses both left and right context.
- A decoder can choose to *attend* to different elements of the encoded input. This shortcut avoids long distances between encoded and decoded elements.
- Attention uses a *query vector* to compute a distribution over a *key vectors*, using the softmax function.
- Encoding, decoding, and attention are general ideas. Choose architectures that are appropriate to input and output types.

**Next lecture.** Transformers. Then we'll have covered most of the key modeling ideas in the course!

Q: the relation between dependency and attention?

A: attention (i.e. self-attention solve problem inside the sentence like verb is related its subject) dependency.

Neubig, Graham. 2017. Neural machine translation and sequence-to-sequence models: A tutorial. ArXiv:1703.01619.

