Encoder-Decoder Models for Neural Machine Translation

Erweiterungsmodul: Machine Translation
Sommersemester 2023

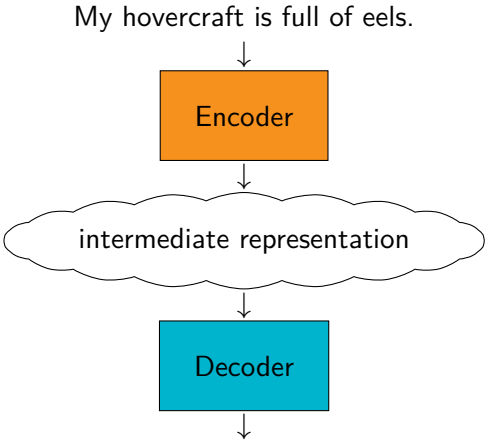
Katharina Hämmerl
haemmerl@cis.lmu.de

Slides adapted from Jindřich Libovický
LMU München, Center for Information and Language Processing

June 6th, 2023

Plan for Today

Conceptual Scheme of an Encoder-Decoder Model



Encoder-Decoder: (Neural) model that takes in a sequence of discrete symbols and generates another sequence.

- Pre-process source (tokenise)
- Convert input into vocabulary indices
- Run the encoder to get an intermediate representation (vector/matrix)
- Run the decoder to get logits ($\sim$ probabilities of tokens)
- Generate vocabulary indices
- Post-process the output (detokenise)

Lecture Overview

- Define Language Models (and Decoders)
- Explaining LMs with RNNs
- What the Encoder is For
- Explain (Cross-)Attention in RNNs
- Intuitions of Attention
- Encoder vs Encoder-Decoder vs Decoder-only

Decoders are Language Models

Language Model: Definition

LM = estimator of sentence probabilities given its (pre-)training corpus

- From now on: sentence = sequence of words $w_1, \dots, w_n$
- Factorize the probability by word
  i.e., no grammar, no hierarchical structure

$$\begin{aligned}\Pr(w_1, \dots, w_n) &= \Pr(w_1) \cdot \Pr(w_2|w_1) \cdot \Pr(w_3|w_2, w_1) \cdot \cdots \\ &= \prod_i^n \Pr(w_i|w_{i-1}, \dots, w_1)\end{aligned}$$

Language Model: Uses

- Intuition: Scores "goodness" of sentences in a language
- Used in Automatic Speech Recognition and Statistical MT to select more probable outputs
- Learns distributional statistics via some objective $\rightarrow$ "knows" the language
  - CBOW and Skipgram objectives for word2vec
  - Masked language modelling for BERT
  - Causal (sequential) language modelling for e.g. GPT-2
- Input: Some context vector(s) of surrounding text.
- Can also *condition* on (a representation of) source sentence $\mathbf{x}$

$\rightarrow$ **A neural decoder is a conditional language model**
(hold on to that thought)

What Happens in a Decoder?

- We simplify, go back in history
- "Left-to-right" (sequential) language modelling always predicts the next word, using the previous context

$$\text{Basically: } \mathrm{P}(w_i | w_{i-1}, w_{i-2}, \ldots, w_{i-n})$$

- I'll recall how the simplest n-gram LMs did this
- I'll show how RNNs did this
- From there we get to conditional language modelling,
- and to Attention

n-gram LMs vs. RNN LMs

n-gram
cool from 1990 to 2013

- Markov assumption → history limited to n words
- Transparent: estimated from n-gram counts in a corpus

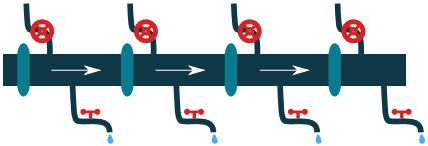
$$\mathrm{P}(w_i|w_{i-1}, w_{i-2}, \ldots, w_{i-n}) \approx \sum_{j=0}^{n} \lambda_j \frac{c(w_i|w_{i-1}, \ldots, w_{i-j})}{c(w_i|w_{i-1}, \ldots, w_{i-j+1})}$$

RNNs
cool from 2013
(to ca. 2018)

- Predict from RNN state—which gathers potentially unlimited history
- Trained by back-propagation to maximise probability of the training data
- Opaque, but works better (as usual with deep learning)

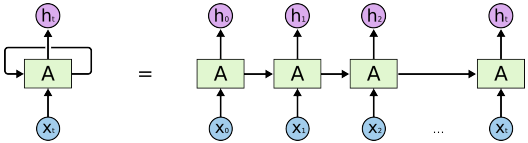
Reminder: Recurrent Neural Networks

RNN = pipeline for information



In every step some information goes in and some information goes out.

Technically: a "`for`" loop applying the same function A on input vectors x_i



At training time unrolled in time: technically just a very deep network

Image on the right: Chris Olah. Understanding LSTM Networks. A blog post: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`

Definitions (for next slide)

embed

Embedding layer. Takes one-hot vector/vocabulary index, returns continuous embedding vector

MLP

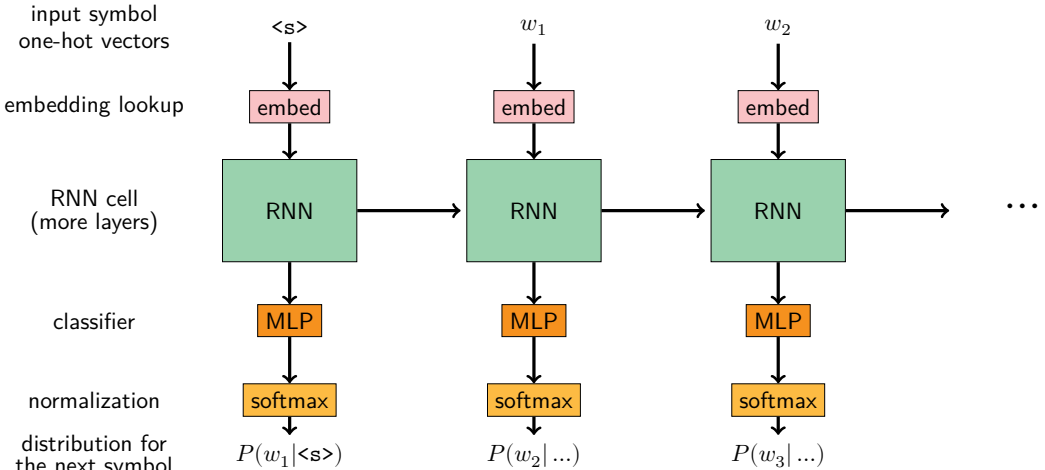
Multi-layer perceptron
n layers, where each does: $\sigma(Wx + b)$
Last layer must output K numbers

softmax

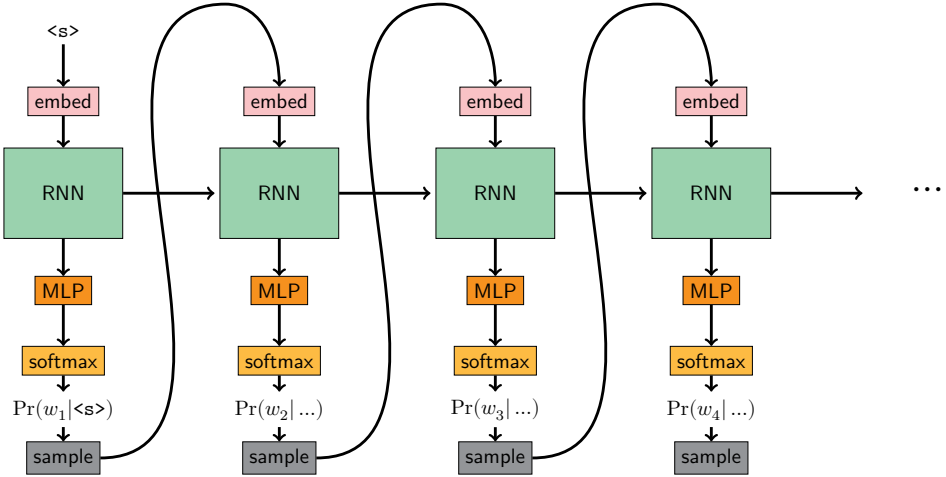
Softmax operation for K classes (vocabulary tokens) with logits $\mathbf{z} = (z_1, \ldots, z_K)$:

$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Modelling an Input Sequence with a Language Model



Getting an Output Sequence from a Language Model



Sampling from a Language Model: Pseudocode

```
last_w = "<s>"
state = initial_state
while last_w != "</s>":
    last_w_embeding = target_embeddings[last_w]
    state = rnn(state, last_w_embeding)
    logits = output_projection(state)
    last_w = vocabulary[np.random.multinomial(1, logits)]
    yield last_w
```

RNN Training

Training objective: negative-log likelihood of the next word(s)

$$\text{NLL} = -\sum_{i}^{n} \log \Pr\left(w_i | w_{i-1}, \ldots, w_1\right)$$

i.e., maximize probability of the correct word.

- Cross-entropy between the predicted distribution and one-hot "true" distribution
- Error from word is backpropagated into the rest of network unrolled in time
- Note: The model only sees "well-behaved" sequences during training, it can break when we sample something weird at inference time → *Exposure bias*

Detour: Why softmax is a good choice

Output layer with softmax (with parameters W, b) — gets categorical distribution:

$$P_y = \text{softmax}(\mathbf{x}) = \Pr(y \mid \mathbf{x}) = \frac{\exp \mathbf{x}^\top W + b}{\sum \exp \mathbf{x}^\top W + b}$$

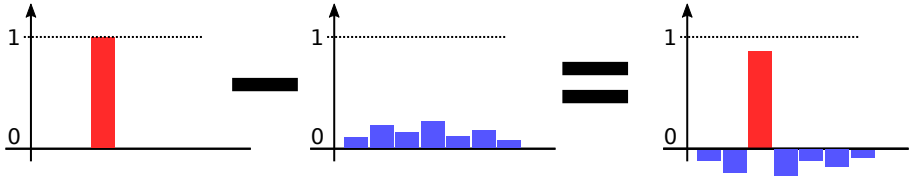
Network error = cross-entropy between estimated distribution and one-hot ground-truth distribution $T = \mathbf{1}(y^*) = (0, 0, \ldots, 1, 0, \ldots, 0)$:

$$\begin{aligned} L(P_y, y^*) = H(P, T) &= -\mathbb{E}_{i \sim T} \log P(i) \\ &= -\sum_i T(i) \log P(i) \\ &= -\log P(y^*) \end{aligned}$$

Derivative of Cross-Entropy

Let $l = \mathbf{x}^\top W + b$, l_{y^*} corresponds to the correct one.

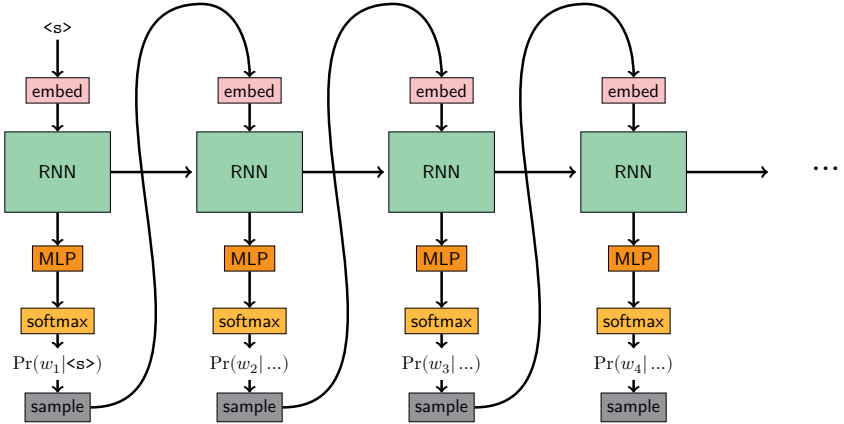
$$\begin{aligned}
\frac{\partial L(P_y, y^*)}{\partial l} &= -\frac{\partial}{\partial l} \log \frac{\exp l_{y^*}}{\sum_j \exp l_j} = -\frac{\partial}{\partial l} l_{y^*} - \log \sum \exp l \\
&= \mathbf{1}_{y^*} + \frac{\partial}{\partial l} - \log \sum \exp l = \mathbf{1}_{y^*} - \frac{\sum \mathbf{1}_{y^*} \exp l}{\sum \exp l} = \\
&= \mathbf{1}_{y^*} - P_y(y^*)
\end{aligned}$$



Interpretation: Reinforce the correct logit, suppress the rest.

Conditioning the Language Model & Attention

Again: Getting an Output Sequence from a Language Model



But we are translating, so where is the input sentence?

Conditional Language Model

Formally, we simply condition the distribution of

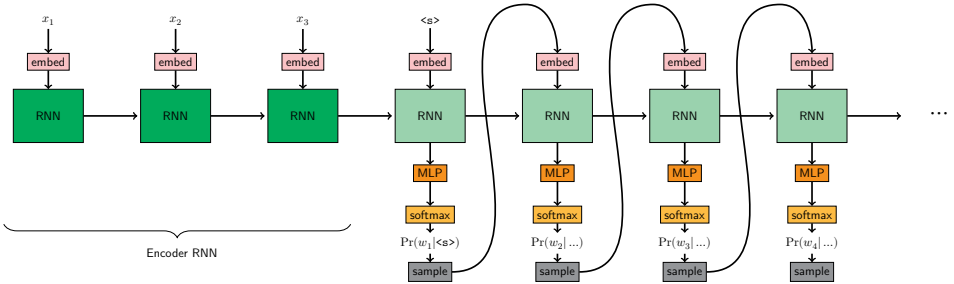
the target sequence $\mathbf{y} = (y_1, \dots, y_{T_y})$

on the source sequence $\mathbf{x} = (x_1, \dots, x_{T_x})$

$$\Pr(y_1, \dots, y_n | \mathbf{x}) = \prod_{i}^{n} \Pr(y_i | y_{i-1}, \dots, y_1, \mathbf{x})$$

We need an *encoder* to get a representation of $\mathbf{x}$!

Vanilla Sequence-to-Sequence Model



- This is a simple encoder-decoder setup: Just continue the RNN
- Interface between encoder and decoder is a single vector, regardless of sentence length

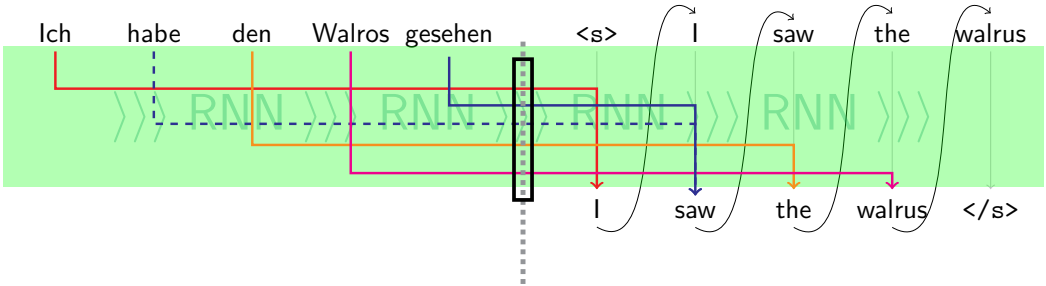
Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112, Montreal, Canada, December 2014

Vanilla Seq2Seq: Pseudocode

```
state = np.zeros(rnn_size)
for w in input_words:
    input_embedding = source_embeddings[w]
    state = enc_cell(encoder_state, input_embedding)

last_w = "<s>"
while last_w != "</s>":
    last_w_embeding = target_embeddings[last_w]
    state = dec_cell(state, last_w_embeding)
    logits = output_projection(state)
    last_w = vocabulary[np.argmax(logits)]
    yield last_w
```

Information Bottleneck



All information runs through this bottleneck

A single vector must represent the entire source sentence

→ **Main weakness, and the motivation for introducing Attention**

Attention Models (1)

- Motivation: We would like to use variable length input representation
- And to get information only from the words we need at any given point

> **Attention** = probabilistic retrieval of encoder states for estimating probability of a target word

- Many different formulations
- Today I'll show Bahdanau et al. (2015), aka MLP Attention
- You'll hear about Scaled Dot-Product Attention (Vaswani et al., 2017) later

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015

Attention Models (2)

You'll probably hear about these terms:

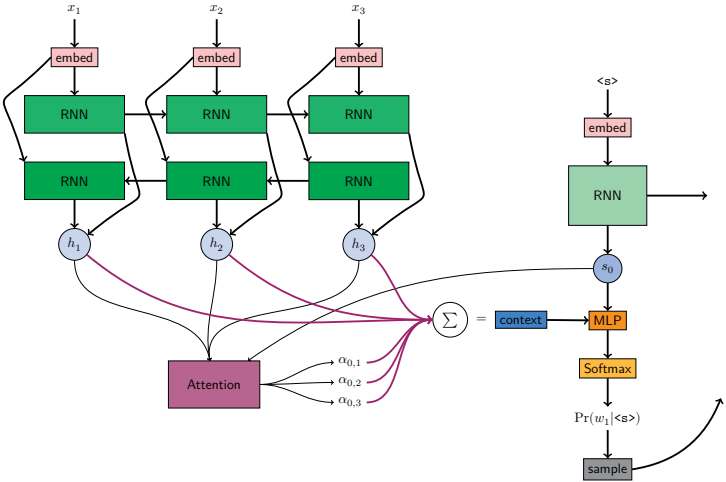
- **Query** = hidden state of the decoder at timestep i
- **Values** = encoder hidden states
- **Keys** = also encoder hidden states, but with a different W

This comes from the *retrieval* metaphor $\rightarrow$ finding matching encoder states for the given decoder state.

Note: These terms don't actually appear in Bahdanau et al. (2015)! They started to be used later.

But: **Query** and **Values** have analogues in this paper.

Sequence-to-Sequence Model With Attention



- Decoder step starts as usual state s_0 ≈ retrieval **query**
- Encoder = bidirectional RNN states h_i ≈ **values**
- Goal: Weighted average of encoder states = context
- Computing weights $\alpha_{i,j}$ = Attention
- Concatenate s_0 & context
- MLP + Softmax predict next word

Attention Model in Equations (1)

Inputs:

decoder state s_i

encoder states $h_j = \left[\overrightarrow{h_j}; \overleftarrow{h_j}\right] \quad \forall i = 1 \dots T_x$

Attention energies:

$$e_{ij} = v_a^\top \tanh\left(W_a s_{i-1} + U_a h_j + b_a\right)$$

Attention distribution:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

Softmax!

Context vector:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Attention Model in Equations (2)

Output projection:

$$t_i = \text{MLP}\left(s_{i-1} \oplus v_{y_{i-1}} \oplus c_i\right)$$

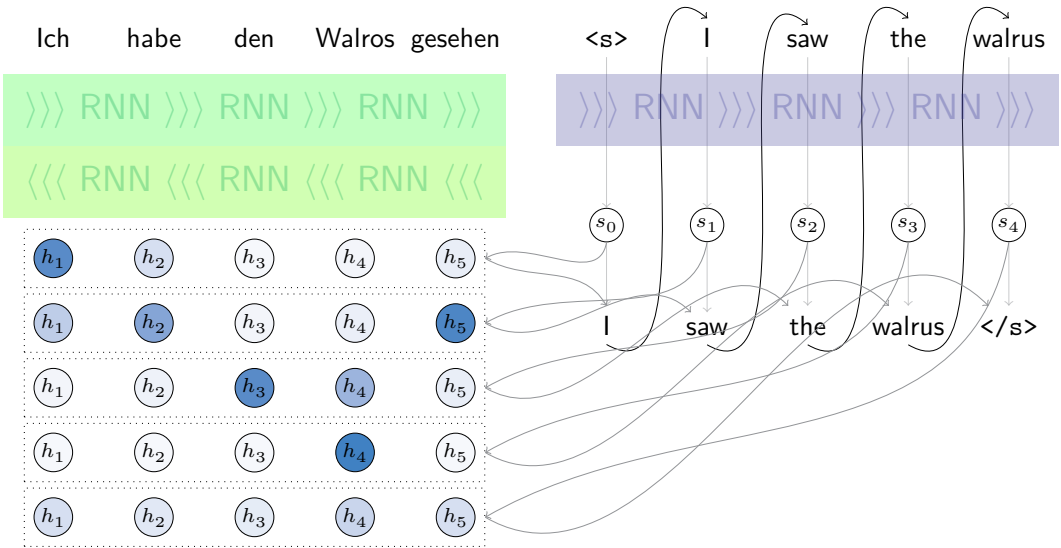
...attention is mixed with the hidden state

Output distribution:

$$p\left(y_i = k | s_i, y_{i-1}, c_i\right) \propto \exp\left(W_o t_i + b_k\right)_k$$

- Different version of attentive decoders exist
- Alternative: keep the context vector as input for the next step
- Multilayer RNNs: attention between/after layers

Workings of the Attentive Seq2Seq model



Seq2Seq with attention: Pseudocode (1)

```
state = np.zeros(emb_size)
fw_states = []
for w in input_words:
    input_embedding = source_embeddings[w]
    state, _ = fw_enc_cell(encoder_state, input_embedding)
    fw_states.append(state)

bw_states = []
state = np.zeros(emb_size)
for w in reversed(input_words):
    input_embedding = source_embeddings[w]
    state, _ = bw_enc_cell(encoder_state, input_embedding)
    bw_states.append(state)

enc_states = [np.concatenate(fw, bw) for fw, bw in zip(fw_states,
    reversed(bw_states))]
```

Seq2Seq with attention: Pseudocode (2)

```
last_w = "<s>"
while last_w != "</s>":
    last_w_embeding = target_embeddings[last_w]
    state = dec_cell(state, last_w_embeding)
    alphas = attention(state, enc_states)
    context = sum(a * state for a, state in zip(alphas, enc_states))
    logits = output_projection(np.concatenate(state, context,
        last_w_embeding))
    last_w = np.argmax(logits)
    yield last_w
```

Attention Visualization (1)

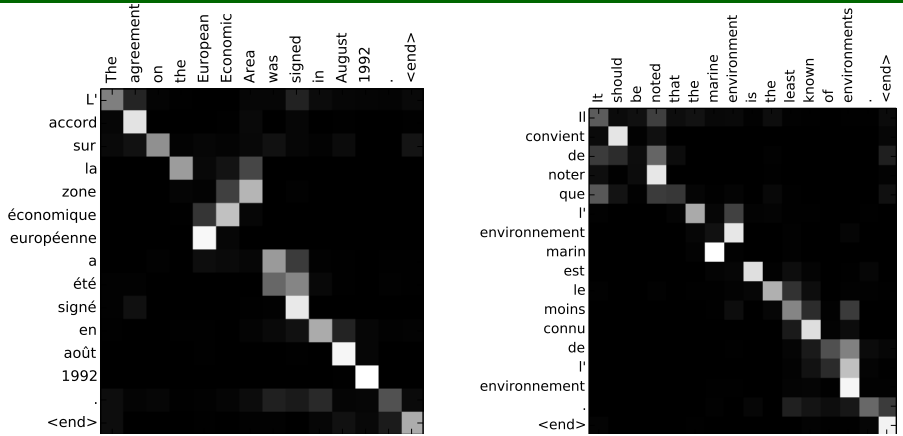


Image from: Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015

Attention Visualization (2)

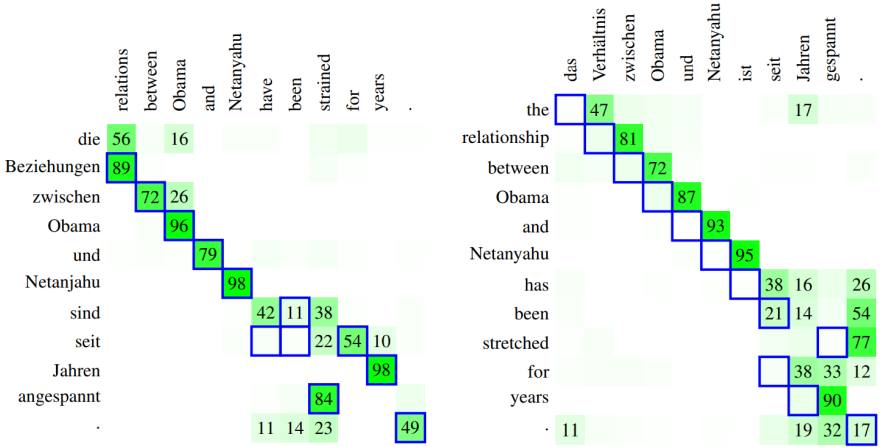


Image from: Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, Canada, August 2017. Association for Computational Linguistics

Attention vs. Alignment

Differences between attention model and word alignment used for phrase table generation:

attention (NMT)	**alignment (SMT)**
probabilistic	discrete
used “freely”	used to build phrase tables
LM generates	LM discriminates

Image Captioning

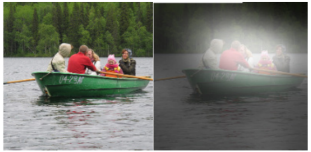
Attention over CNN for image classification:

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

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

A little girl sitting on a bed with a teddy bear.

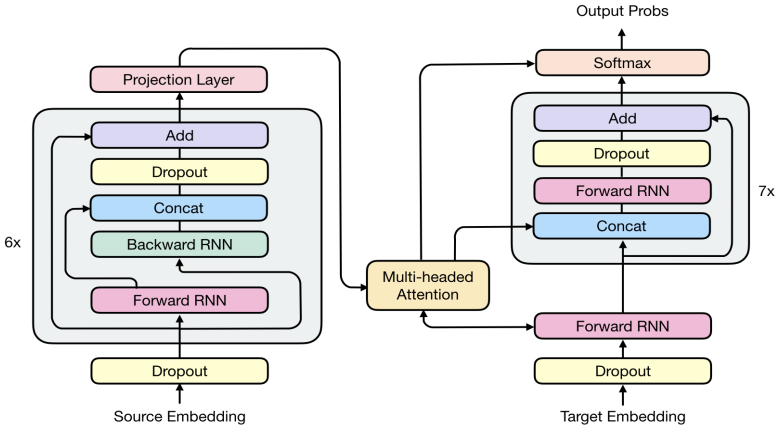


A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Image from: Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France, July 2015. PMLR

Further Architecture Tricks



Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86, Melbourne, Australia, July 2018. Association for Computational Linguistics

Training Seq2Seq Model

Optimize negative log-likelihood of parallel data,
backpropagation does the rest.

...if you choose the right optimizer, learning rate, model hyperparameters, prepare data, do back-translation, monolingual pre-training...

Inference

Getting output (1)

- Encoder-decoder is a conditional language model
- For a pair $\mathbf{x}$ and $\mathbf{y}$, we can compute:

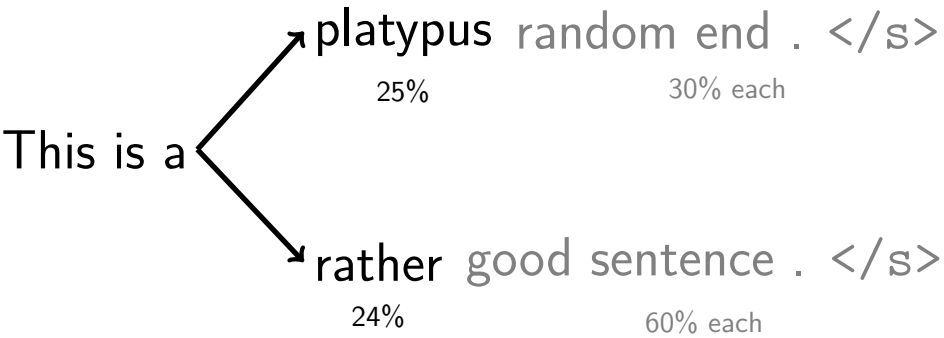
$$\Pr(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{T_y} \Pr(y_i|\mathbf{y}_{:i}, \mathbf{x})$$

- When decoding we want to get

$$\mathbf{y}^* = \underset{\mathbf{y}'}{\operatorname{argmax}} \Pr(\mathbf{y}'|x)$$

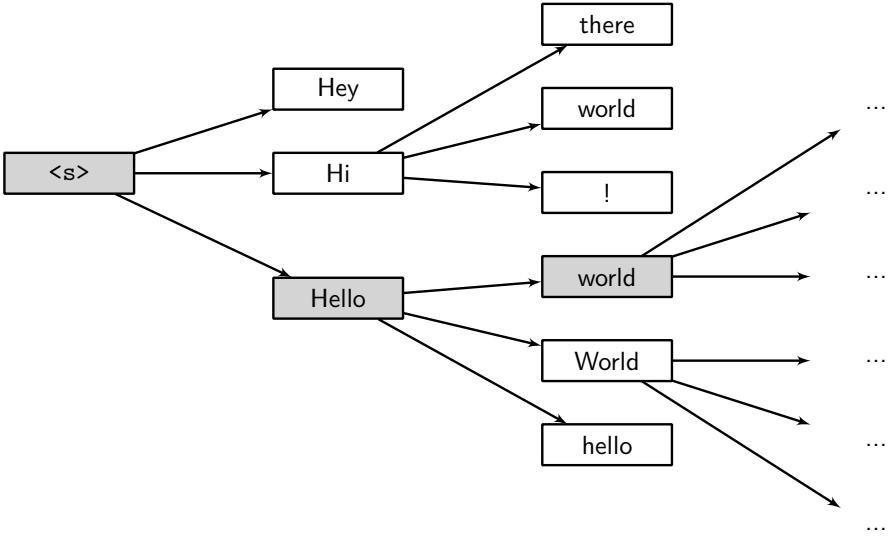
$\rightarrow$ Recall: Greedy Decoding vs Beam Search!

Greedy Decoding: A Silly Example



⚠ Greedy decoding can easily miss the best option. ⚠

Beam Search: Example



Implementation Details

- Multiplying of too many small numbers $\rightarrow$ float underflow
  need to compute in log domain and add logarithms
- Sentences can have different lengths

This	is	a	good	long	sentence	.	</s>	
0.7	$\times$ 0.6	$\times$ 0.9	$\times$ 0.1	$\times$ 0.4	$\times$ 0.4	$\times$ 0.8	$\times$ 0.9	**= 0.004**

This	</s>	
0.7	$\times$ 0.01	**= 0.007**

  $\Rightarrow$ use the geometric mean instead of probabilities directly
- Sorting candidates is expensive, asymptotically $|V| \log |V|$:
  k-best can be found in linear time, $|V| \sim 10^4 - 10^5$

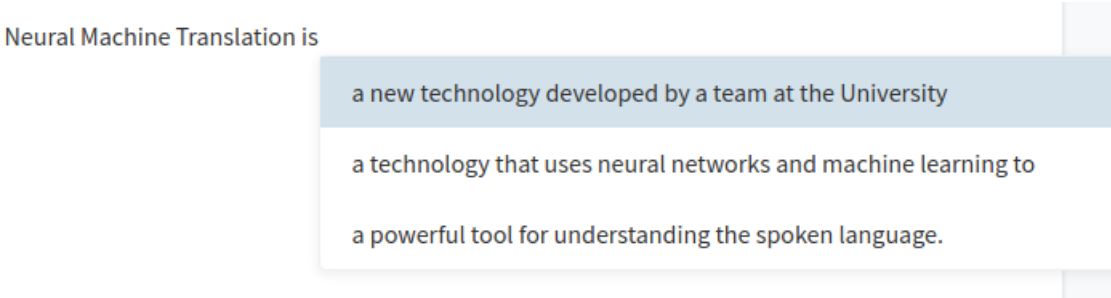
Encoder vs. Encoder-Decoder vs. Decoder-only Models

Encoder-Only Models

- Can pre-train the encoder on monolingual source language data
- But we can also use pre-trained encoder representations for other tasks: classification, sequence labelling, semantic similarity,...

  $\rightarrow$ e.g., BERT!
- This in turn expanded to multilingual encoders (which you'll also hear some more about)

What about Generative Language Models?



(Example from GPT-2, a Tranformer-based English language model, screenshot from `https://transformer.huggingface.co/doc/gpt2-large`)

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog.*, 2019

The GPTs are *decoder-only* models.

Getting output(2)

Note:

- Same general principle for generating in decoder-only language models
- However: More emphasis on *sampling* strategies for generative models
- MT: Find best translation for a given input. There may be multiple good translations, but there should be relatively little variation
- Generative models: No source sentence; there are many good continuations, and we want *diverse* outputs $\rightarrow$ sample from the output distribution with certain constraints
- For example: Temperature (control "how random" sampling is), top-k-sampling (sample from top k suggestions), top-p-sampling (sample from top p% of probability mass). Typically combined with beam search.
- There's more!

Summary and References

Some History (1)

- **2013** First encoder-decoder model

  Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October 2013. Association for Computational Linguistics

- **2014** First really usable encoder-decoder model

  Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112, Montreal, Canada, December 2014

- **2014/2015** Added attention (crucial innovation in NLP)

  Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015

- **2016/2017** WMT winners used RNN-based neural systems

  Rico Sennrich, Barry Haddow, and Alexandra Birch. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376, Berlin, Germany, August 2016. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W16-2323

Some History (2)

- **2017** Transformers invented (outperformed RNN)

  Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 6000–6010, Long Beach, CA, USA, December 2017. Curran Associates, Inc

- **2018** RNMT+ architecture shows that RNNs can be on par with Transformers

  Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86, Melbourne, Australia, July 2018. Association for Computational Linguistics

The development still goes on...

Unsupervised models, document context, non-autoregressive models, multilingual models, ...

Summary

- Encoder-decoder architecture = major paradigm in MT
- Encoder-decoder architecture = conditional language model
- Attention = way of conditioning the decoder on the encoder
- Attention = probabilistic vector retrieval
- We model probability, but need heuristics to get a good sentence from the model