Encoder-Decoder Models for Neural Machine Translation

Erweiterungsmodul: Machine Translation
Sommersemester 2020

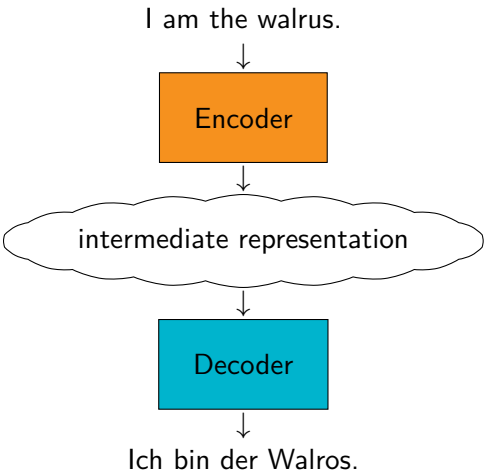
Jindřich Libovický
libovicky@cis.lmu.de

LMU München, The Center for Information and Language Processing

May 27, 2020

Model Concept

Conceptual Scheme of the Model



Neural model with a sequence of discrete symbols as an input that generates another sequence of discrete symbols as an output.

- pre-process source sentence (tokenize, split into smaller units)
- convert input into vocabulary indices
- run the encoder to get an intermediate representation (vector/matrix)
- run the decoder
- postprocess the output (detokenize)

From Statistical to Neural MT

Statistical

- Meaning of a sentence can be decomposed into phrases.
- Adequacy and fluency can be modeled separately.

Neural

- Sentences are sequence of words (or smaller discrete units).
- (And maybe something about words having independent meaning and compositionality, but this is difficult.)

Neural Model are Black Boxes

⚠ Disclaimer ⚠

- Don't trust much the intuitions presented there: NNs can learn things different than we assume.
- All concepts used are from one half technical concepts, from the other half lose metaphors.

Language Models and Decoders

What is a Language Model

LM = an estimator of a sentence probability given a language

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

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

What is it good for?

- Substitute for grammar: tells what is a good sentence in a language
- Used in ASR, and statistical MT to select more probable outputs
- Being able to predict next word = knowing the language
  - language modeling is training objective for word2vec
  - BERT is a masked language model

- **Neural decoder is a conditional language model.**

n-gram vs. neural LMs

n-gram
cool from 1990 to 2013

- Limited history = Markov assumption
- Transparent: estimated from n-gram counts in a corpus

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

Neural
cool since 2013

- Conditioned on RNN state which gather potentially unlimited history
- Trained by back-propagation to maximize probability of the training data
- Opaque, but works better (as usual with deep learning)

Sequence Labeling

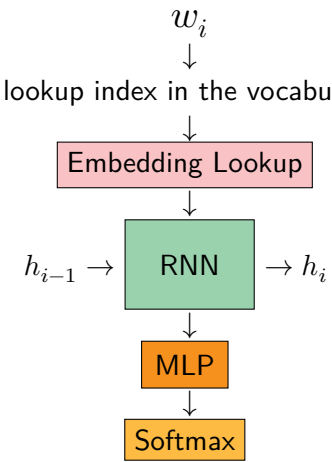
- Assign a label to each word in a sentence.
- Tasks formulated as sequence labeling:
  - Part-of-Speech Tagging
  - Named Entity Recognition
  - Filling missing punctuation

MLP = Multilayer perceptron

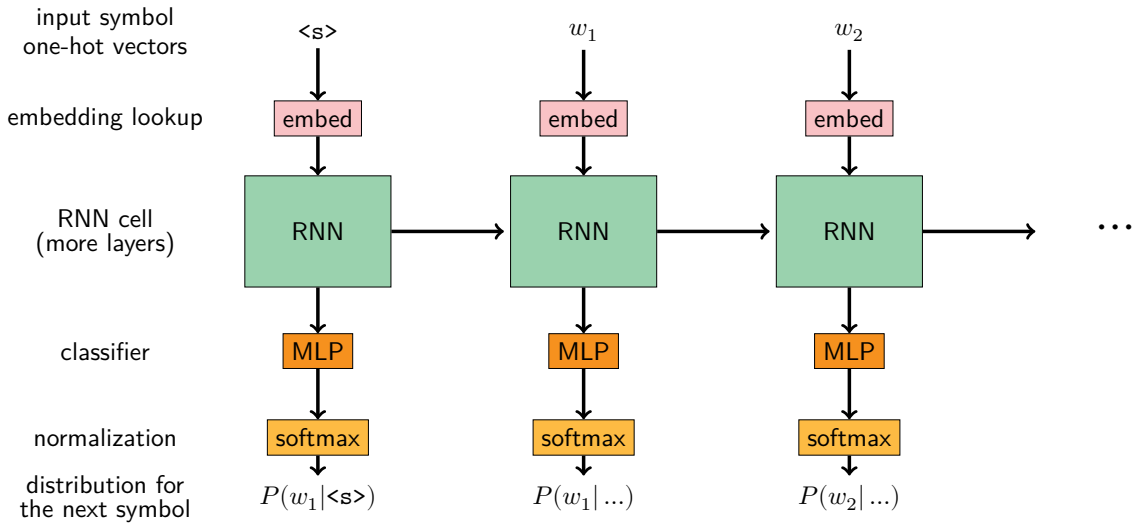
$n\times$ layer: $\sigma\left(Wx + b\right)$

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

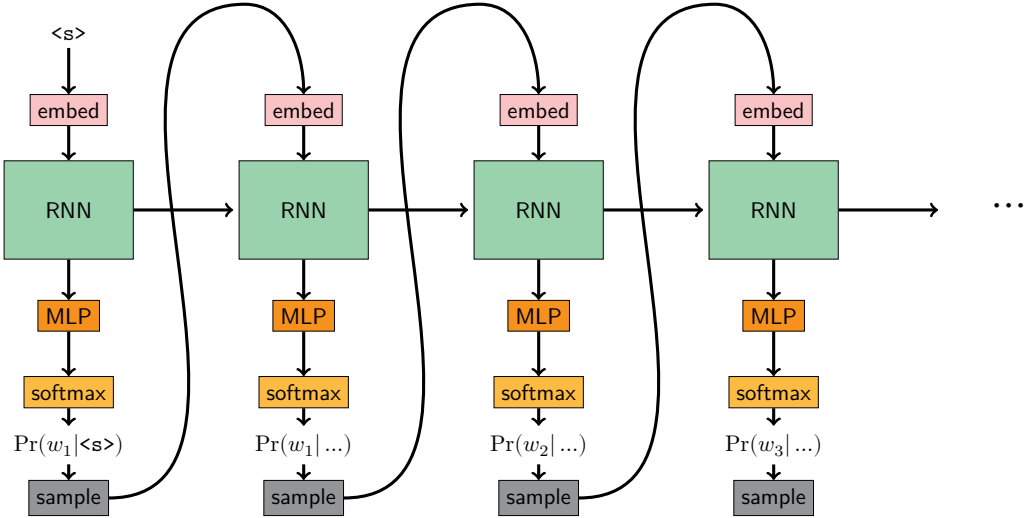
$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$



Language Model as Sequence Labeling



Sampling 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.multimial(1, logits)]
    yield last_w
```

Training

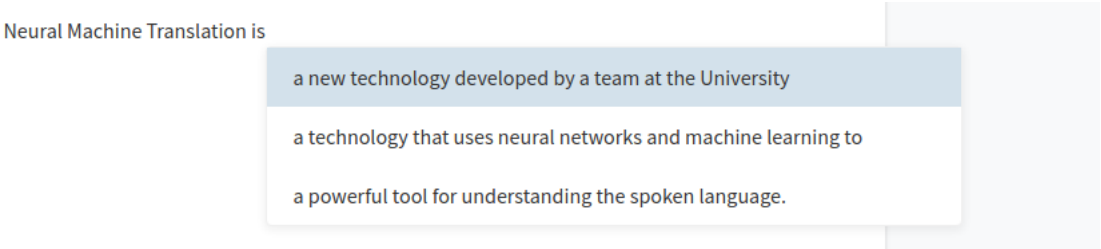
Training objective: negative-log likelihood:

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

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
- Prone to exposure bias: during trainining only well-behaved sequences, it can break when we sample something weird at inference time

Generating from a Language Model



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

Where is the source language?

Conditioning the Language Model & Attention

Conditional Language Model

Formally it is simple, condition distribution of

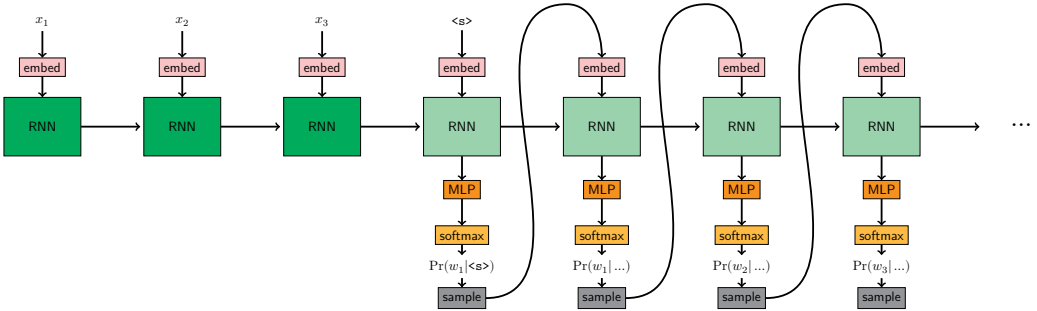
- target sequence $\mathbf{y} = (y_1, \ldots, y_{T_y})$ on
- source sequence $\mathbf{x} = (x_1, \ldots, x_{T_x})$

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

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

What about just continuing an RNN...

Sequence-to-Sequence Model



- The interface between encoder and decoder is a single vector regardless the 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. Curran Associates, Inc

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
```

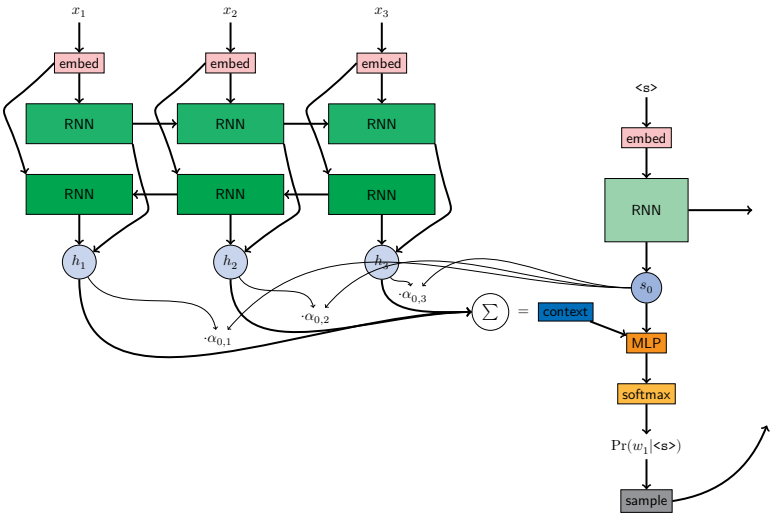
The Attention Model

- Motivation: It would be nice to can use variable length input representation
- RNN returns one state per word ...
- ...what if we were able to get only information from words we need to generate a word.

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

Query = hidden states of the decoder
Values = encoder hidden states

Sequence-to-Sequence Model With Attention



- Encoder = bidirectional RNN
- Decoder step starts as usual
- Decoder state s_0 used to compute distribution the over encoder states
- Weighted average of encoder states = context vector
- Decoder state & context concatenated

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. URL http://arxiv.org/abs/1409.0473

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:

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

Context vector:

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

Attention Model in Equations (2)

Output projection:

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

...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

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)

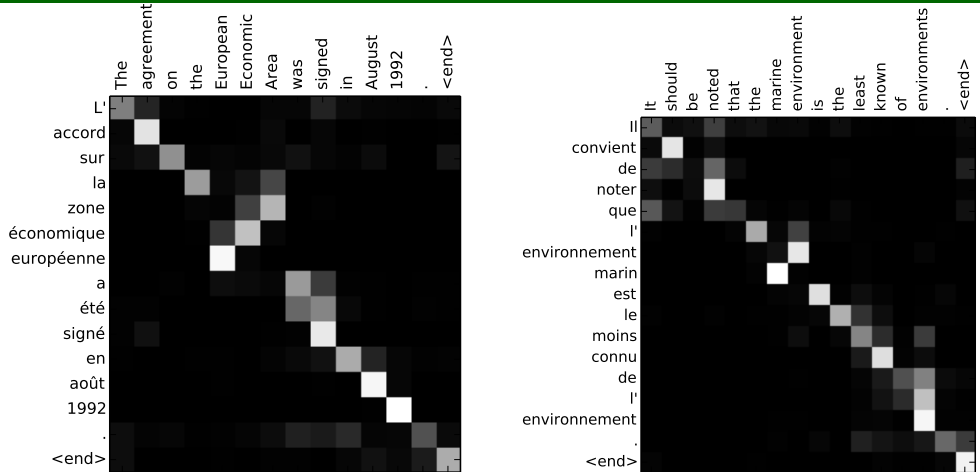


Image from: Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473

Attention Visualization (2)

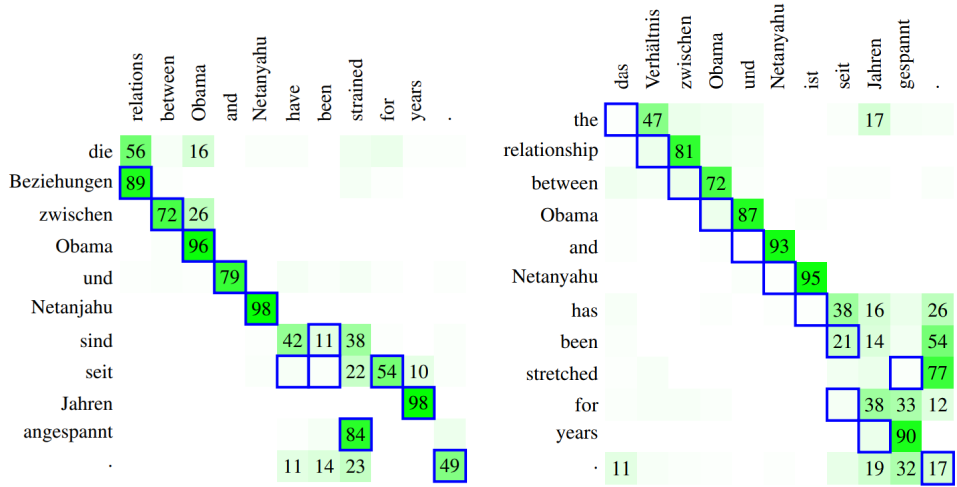


Image from: Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/w17-3204. URL http://dx.doi.org/10.18653/v1/w17-3204

Attention vs. Alignment

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

attention (NMT)	**alignment (SMT)**
probabilistic	discrete
declarative	imperative
LM generates	LM discriminates

Image Captioning

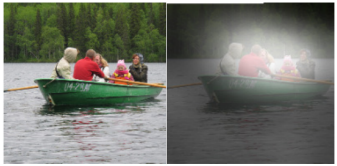
Attention over CNN for image classification:

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.

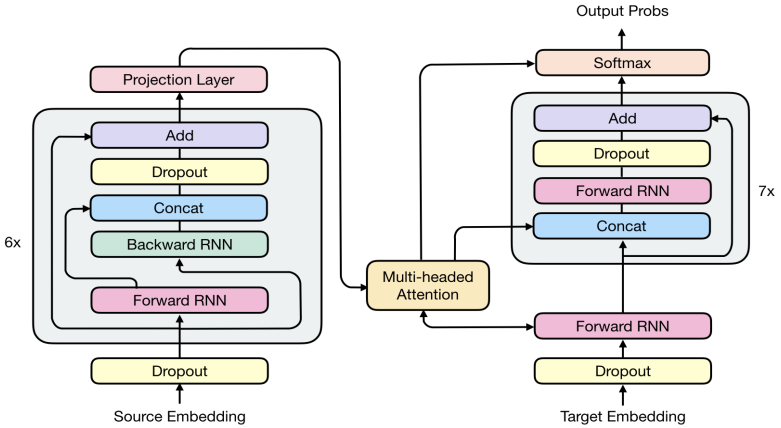


A group of people sitting on a boat in the water.

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 Architectural Tricks



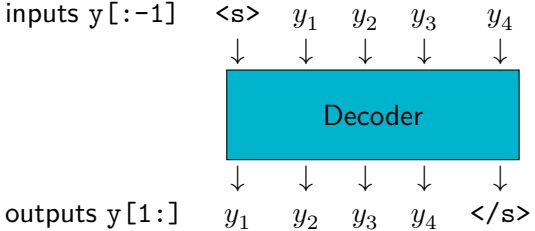
Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine 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. doi: 10.18653/v1/P18-1008. URL https://www.aclweb.org/anthology/P18-1008

Training Seq2Seq Model

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

If you choose a right optimizer, learning rate, model hyper-parameters, prepare data, do back-translation, monolingual pre-training ...

During training confusion: decoder inputs vs. output



Inference

Getting output

- 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

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

☠ **Enumerating all $\mathbf{y}'$s is computationally intractable** ☠

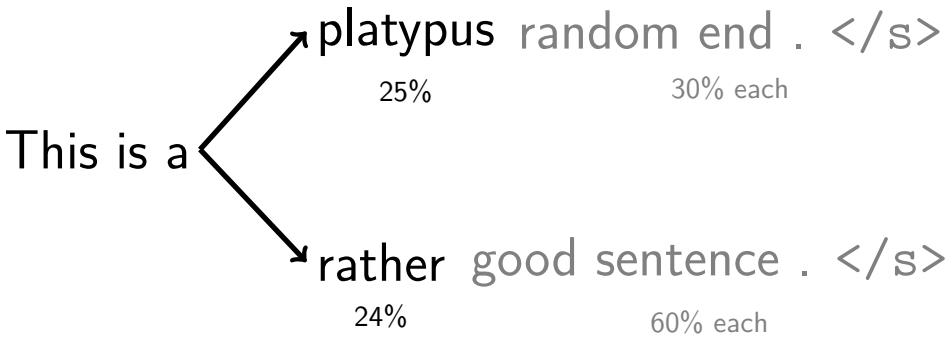
Greedy Decoding

In each step, take the maximum probable word.

$$y_i^* = \underset{y_i}{\operatorname{argmax}} \Pr\left(y_i | y_{i-1}^*, \ldots, \texttt{<s>}\right)$$

```
last_w = "<s>"
state = initial_state
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
```

What if...



⚠ Greedy decoding can easily miss the best option. ⚠

Beam Search

Keep a small k of hypothesis (typically 10).

1. Begin with a single empty hypothesis in the beam.
2. In each time step:
   1. Extend all hypotheses in the beam by all (or the most probable) from the output distribution (we call these *candidate hypotheses*)
   2. Score the candidate hypotheses and add them to the beam
3. Finish if all k-best hypotheses end with </s>
4. Sort the hypotheses by their score and output the best one.

Beam Search: Pseudocode

```
beam = [(["<s>"], initial_state, 1.0)]
while any(hyp[-1] != "</s>" for hyp, _, _ in beam):
    candidates = []

    for hyp, state, score in beam:
    distribution, new_state = decoder_step(hyp[-1], state, encoder_states)
        for i, prob in enumerate(distribution):
            candidates.append(hyp + [vocabulary[i]], new_state, score * prob)

    beam = take_best(k, candidates)
```

Implementation issues

- 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**

- Sorting candidates is expensive: k-best can be found in linear time

Final Remarks

A bit of 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. URL https://www.aclweb.org/anthology/D13-1176

- **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. Curran Associates, Inc

- **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. URL http://arxiv.org/abs/1409.0473

- **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. doi: 10.18653/v1/W16-2323. URL https://www.aclweb.org/anthology/W16-2323

A bit of 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. doi: 10.18653/v1/P18-1008. URL https://www.aclweb.org/anthology/P18-1008

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