Attentive Sequence-to-Sequence Learning

March 6, 2018

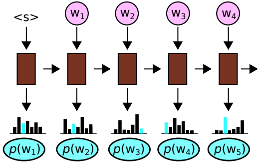
Jindřich Helcl, Jindřich Libovický

Charles Univeristy in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

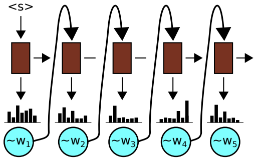
Neural Network Language Models

RNN Language Model

- Train RNN as classifier for next words (unlimited history)



- Can be used to estimate sentence probability / perplexity $\rightarrow$ defines a distribution over sentences
- We can sample from the distribution



Two views on RNN LM

- RNN is a `for` loop (functional `map`) over sequential data
- All outputs are conditional distributions $\rightarrow$ probabilistic distribution over sequences of words

$$P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | w_{i-1}, \ldots, w_1)$$

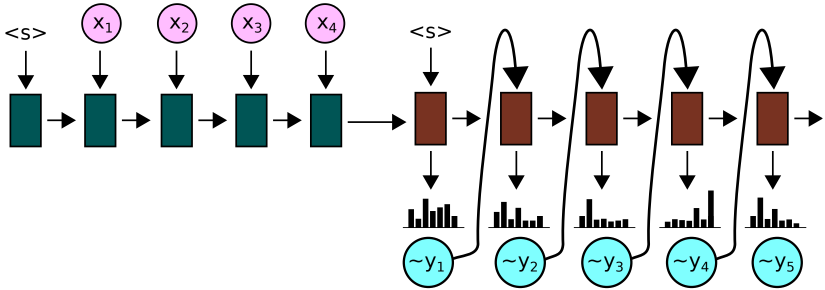
Vanilla Sequence-to-Sequence Model

Encoder-Decoder NMT

- Exploits the conditional LM scheme
- Two networks
    1. A network processing the input sentence into a single vector representation (*encoder*)
    2. A neural language model initialized with the output of the encoder (*decoder*)

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.

Encoder-Decoder – Image



Source language input + target language LM

Encoder-Decoder Model – Code

```
state = np.zeros(emb_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_output = dec_cell(state,
                                 last_w_embeding)
    logits = output_projection(dec_output)
    last_w = np.argmax(logits)
    yield last_w
```

Encoder-Decoder Model – Formal Notation

Data

input embeddings (source language)	$\mathbf{x} = (x_1, \ldots, x_{T_x})$
output embeddings (target language)	$\mathbf{y} = (y_1, \ldots, y_{T_y})$

Encoder

initial state	$h_0 \equiv \mathbf{0}$
j-th state	$h_j = \mathrm{RNN}_{\mathrm{enc}}(h_{j-1}, x_j)$
final state	h_{T_x}

Decoder

initial state	$s_0 = h_{T_x}$
i-th decoder state	$s_i = \mathrm{RNN}_{\mathrm{dec}}(s_{i-1}, \hat{y}_i)$
i-th word score	$t_{i+1} = U_o + V_o E y_i + b_o$, or multi-layer projection
output	$\hat{y}_{i+1} = \arg\max t_{i+1}$

Encoder-Decoder: Training Objective

For output word y_i we have:

- Estimated conditional distribution $\hat{p}_i = \frac{\exp t_i}{\sum \exp t_i}$ (softmax function)
- Unknown true distribution p_i, we lay $p_i \equiv \mathbf{1}\left[y_i\right]$

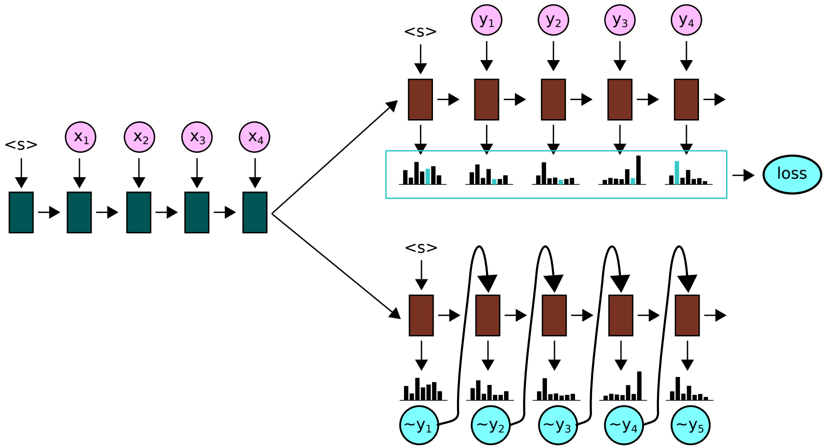
Cross entropy $\approx$ distance of $\hat{p}$ and p:

$$\mathcal{L} = H(\hat{p}, p) = \mathbf{E}_p\left(-\log \hat{p}\right) = -\log \hat{p}(y_i)$$

...computing $\frac{\partial \mathcal{L}}{\partial t_i}$ is super simple

Implementation: Runtime vs. training

runtime: $\hat{y}_j$ (decoded) $\times$ **training:** y_j (ground truth)



Sutskever et al.

- Reverse input sequence
- Impressive empirical results – made researchers believe NMT is way to go

Evaluation on WMT14 EN $\rightarrow$ FR test set:

method	BLEU score
vanilla SMT	33.0
tuned SMT	37.0
Sutskever et al.: reversed	30.6
–"–: ensemble + beam search	34.8
–"–: vanilla SMT rescoring	36.5
Bahdanau's attention	28.5

Why is better Bahdanau's model worse?

Sutskever et al. × Bahdanau et al.

	Sutskever et al.	**Bahdanau et al.**
vocabulary	160k enc, 80k dec	30k both
encoder	4× LSTM, 1,000 units	bidi GRU, 2,000
decoder	4× LSTM, 1,000 units	GRU, 1,000 units
word embeddings	1,000 dimensions	620 dimensions
training time	7.5 epochs	5 epochs

With Bahdanau's model size:

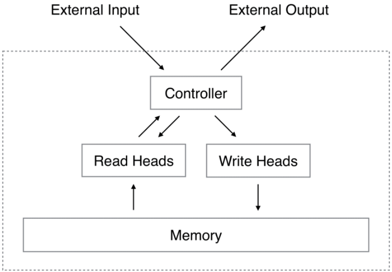
method	BLEU score
encoder-decoder	13.9
attention model	28.5

Attentive Sequence-to-Sequence Learning

Main Idea

- Same as reversing input: do not force the network to catch long-distance dependencies
- Use decoder state only for target sentence dependencies and a as query for the source word sentence
- RNN can serve as LM — it can store the language context in their hidden states

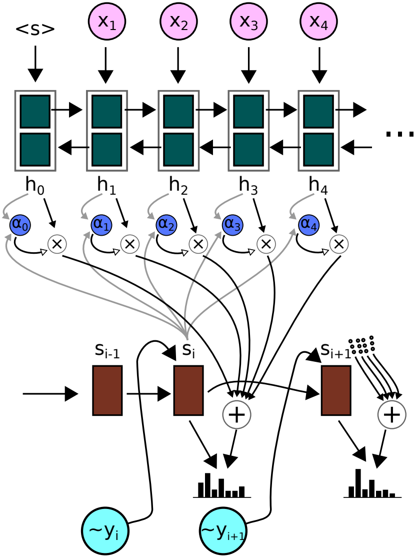
Inspiration: Neural Turing Machine



- General architecture for learning algorithmic tasks, finite imitation of a Turing Machine
- Needs to address memory somehow – either by position or by content

- In fact does not work well – it hardly manages simple algorithmic tasks
- Content-based addressing $\rightarrow$ attention

Attention Model



Attention Model in Equations (1)

Inputs:

decoder state s_i

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

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:

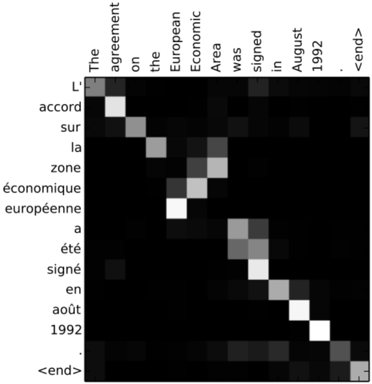
$$t_i = \mathrm{MLP}\left(U_o s_{i-1} + V_o E y_{i-1} + C_o c_i + b_o\right)$$

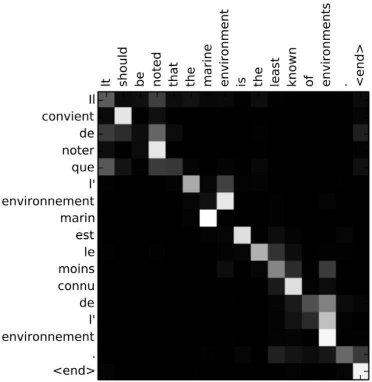
...context vector is mixed with the hidden state

Output distribution:

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

Attention Visualization





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.

Source: Xu, Kelvin, et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." *ICML*. Vol. 14. 2015.

Reading for the Next Week

Vaswani, Ashish, et al. "Attention is all you need." Advances in Neural Information Processing Systems. 2017.
http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Question:

The model uses the scaled dot-product attention which is a non-parametric variant of the attention mechanism. Why do you think it is sufficient in this setup? Do you think it would work in the recurrent model as well?

The way the model processes the sequence is principally different from RNNs or CNNs. Does it agree with your intuition of how language should be processed?