EE-559 – Deep learning

13.1. Attention for Memory and Sequence Translation

François Fleuret
https://fleuret.org/ee559/
Jun 3, 2020

In all the operations we have seen such as fully connected layers, convolutions, or poolings, the contribution of a value in the input tensor to a value in the output tensor is entirely driven by their [relative] locations [in the tensor].

Attention mechanisms aggregate features with an importance score that

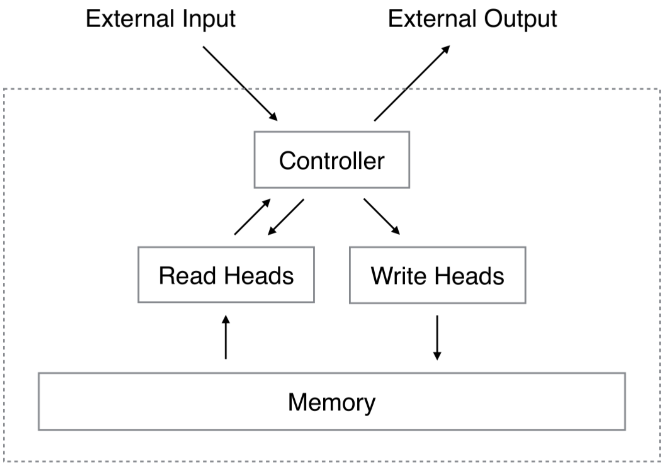
- depends on the feature themselves, not on their positions in the tensor,
- relax locality constraints.

Attention mechanisms modulate dynamically the weighting of different parts of a signal and allow the representation and allocation of information channels to be dependent on the activations themselves.

While they were developed to equip deep-learning models with memory-like modules (Graves et al., 2014), their main use now is to provide long-term dependency for sequence-to-sequence translation (Vaswani et al., 2017).

Neural Turing Machine

Graves et al. (2014) proposed to equip a deep model with an explicit memory to allow for long-term storage and retrieval.



(Graves et al., 2014)

The said module has an hidden internal state that takes the form of a tensor

$$M_t \in \mathbb{R}^{N \times M}$$

where t is the time step, N is the number of entries in the memory and M is their dimension.

A "controller" is implemented as a standard feed-forward or recurrent model and at every iteration t it computes activations that modulate the reading / writing operations.

More formally, the memory module implements

- Reading, where given attention weights $w_t \in \mathbb{R}_+^N$, $\sum_n w_t(n) = 1$, it gets

$$r_t = \sum_{n=1}^{N} w_t(n) M_t(n).$$

- Writing, where given attention weights w_t, an *erase vector* $e_t \in [0,1]^M$ and an *add vector* $a_t \in \mathbb{R}^M$ the memory is updated with

$$\forall n, M_t(n) = M_{t-1}(n)(1 - w_t(n) e_t) + w_t(n) a_t.$$

The controller has multiple "heads", and computes at each t, for each writing head w_t, e_t, a_t, and for each reading head w_t, and gets back a read value r_t.

The vectors w_t are themselves recurrent, and the controller can strengthen them on certain **key values**, and/or shift them.

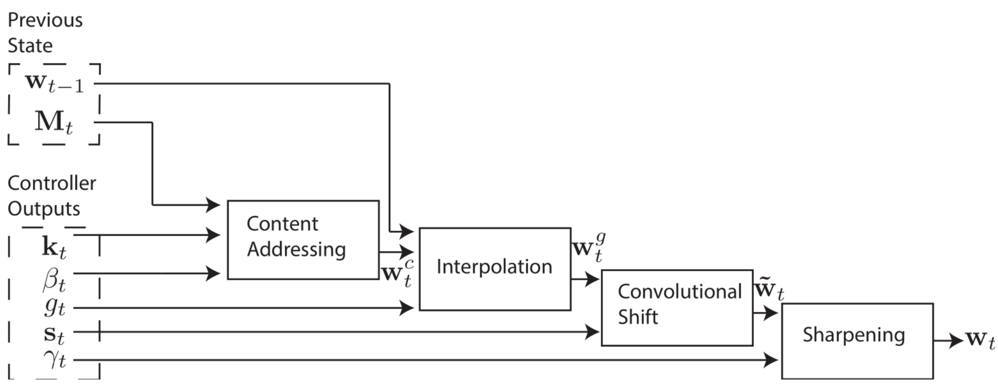
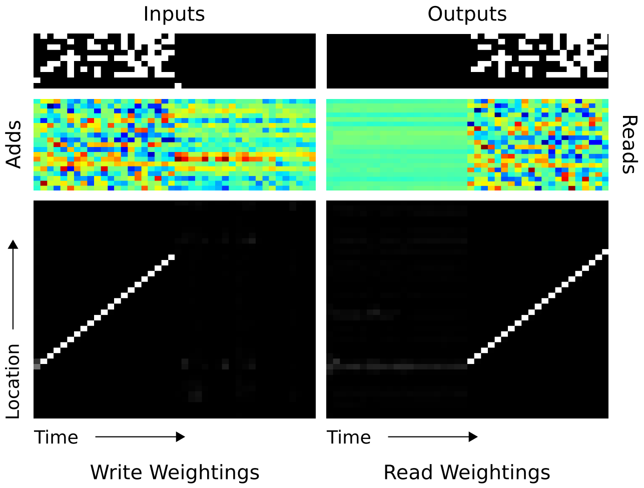
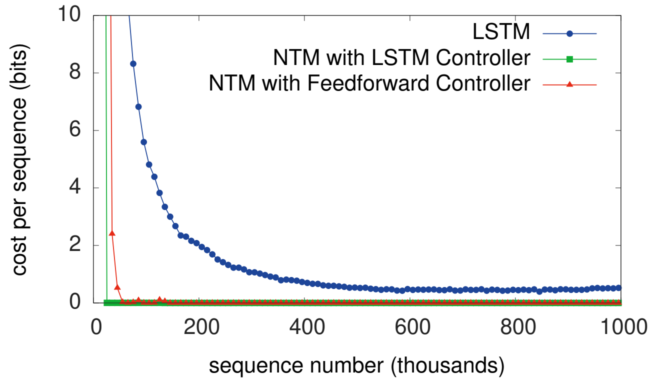


Figure 2: Flow Diagram of the Addressing Mechanism. The *key vector*, $\mathbf{k}_t$, and *key strength*, β_t, are used to perform content-based addressing of the memory matrix, $\mathbf{M}_t$. The resulting content-based weighting is interpolated with the weighting from the previous time step based on the value of the *interpolation gate*, g_t. The *shift weighting*, $\mathbf{s}_t$, determines whether and by how much the weighting is rotated. Finally, depending on γ_t, the weighting is sharpened and used for memory access.

(Graves et al., 2014)

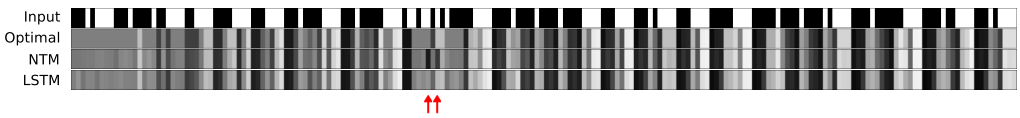
Results on the copy task

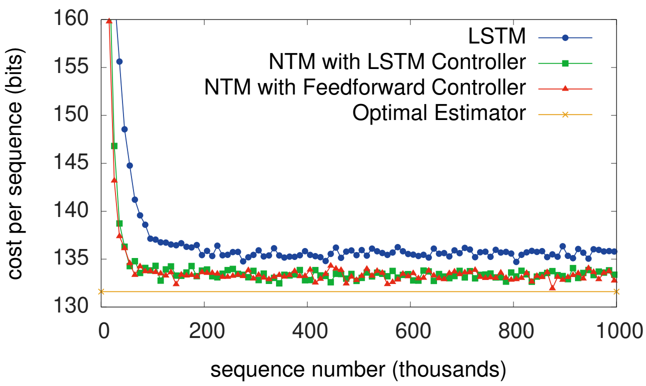




(Graves et al., 2014)

Results on the N-gram task





(Graves et al., 2014)

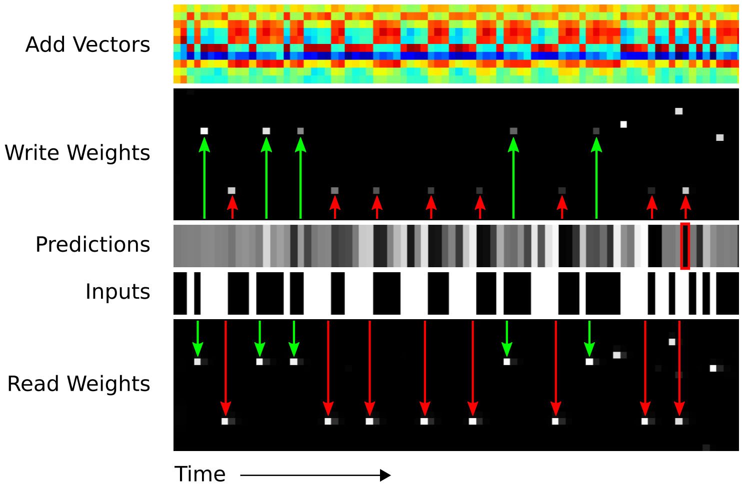


Figure 15: NTM Memory Use During the Dynamic N-Gram Task. The red and green arrows indicate point where the same context is repeatedly observed during the test sequence ("00010" for the green arrows, "01111" for the red arrows). At each such point the same location is accessed by the read head, and then, on the next time-step, accessed by the write head. We postulate that the network uses the writes to keep count of the fraction of ones and zeros following each context in the sequence so far. This is supported by the add vectors, which are clearly anti-correlated at places where the input is one or zero, suggesting a distributed "counter." Note that the write weightings grow fainter as the same context is repeatedly seen; this may be because the memory records a ratio of ones to zeros, rather than absolute counts. The red box in the prediction sequence corresponds to the mistake at the first red arrow in Figure 14; the controller appears to have accessed the wrong memory location, as the previous context was "01101" and not "01111."

(Graves et al., 2014)

Attention for seq2seq

Given an input sequence $x_1, \dots, x_T$, the standard approach for sequence-to-sequence translation (Sutskever et al., 2014) uses a recurrent model

$$h_t = f(x_t, h_{t-1}),$$

and considers that the final hidden state

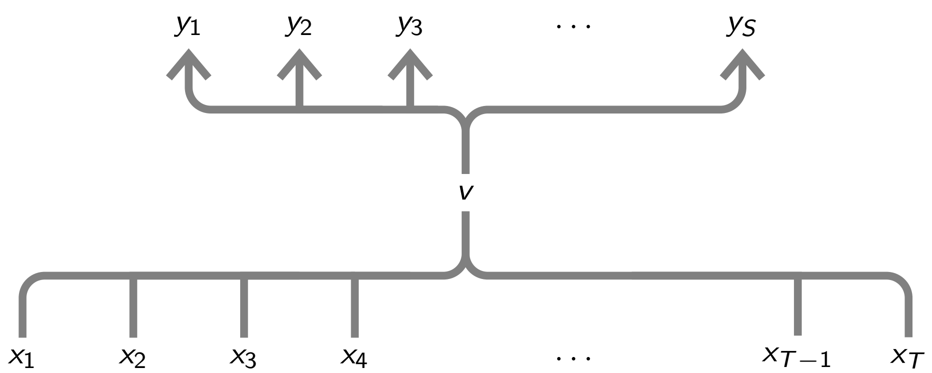
$$v = h_T$$

carries enough information to drive an auto-regressive generative model

$$y_t \sim p\left(y_1, \dots, y_{t-1}, v\right),$$

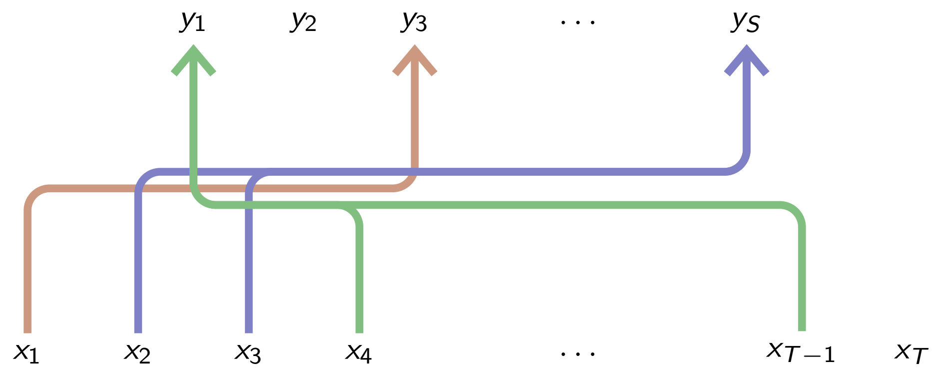
itself implemented with another RNN.

The main weakness of such an approach is that all the information has to flow through a single state v, whose capacity has to accommodate any situation.



There are no direct "channels" to transport local information from the input sequence to the place where it is useful in the resulting sequence.

Attention mechanisms (Bahdanau et al., 2014) can transport information from parts of the signal to other parts **specified dynamically**.



Bahdanau et al. (2014) proposed to extend a standard recurrent model with such a mechanism. They first run a bi-directionnal RNN to get a hidden state

$$h_i = (h_i^{\rightarrow}, h_i^{\leftarrow}),\ i = 1, \dots, T.$$

From this, they compute a new process $s_i,\ i = 1, \dots, T$ which looks at weighted averages of the h_j, where **the weight are functions of the signal.**

Given $y_1, \dots, y_{i-1}$ and $s_1, \dots, s_{i-1}$ first compute an attention

$$\forall j,\ \alpha_{i,j} = \mathrm{softmax}_j\, a(s_{i-1}, h_j)$$

where a is a one hidden layer tanh MLP (this is "additive attention", or "concatenation").

Then compute the **context vector** from the hs

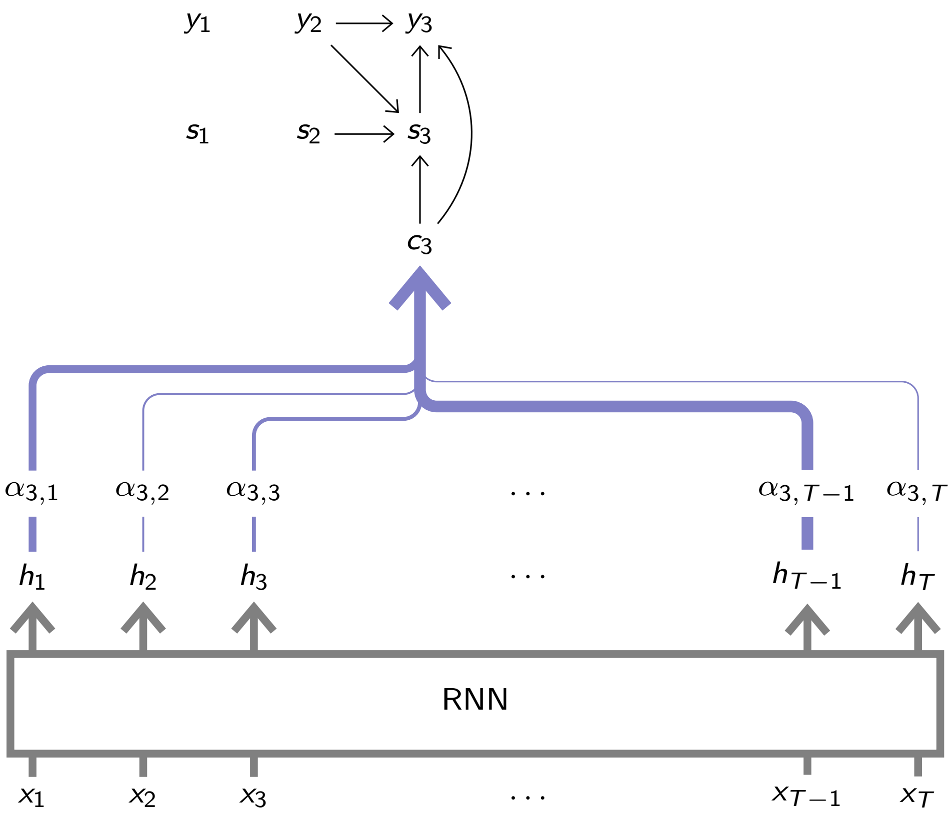
$$c_i = \sum_{j=1}^{T} \alpha_{i,j} h_j.$$

The model can now make the prediction

$$\begin{aligned} s_i &= f(s_{i-1}, y_{i-1}, c_i) \\ y_i &\sim g(y_{i-1}, s_i, c_i) \end{aligned}$$

where f is a GRU (Cho et al., 2014).

This is **context attention** where s_{i-1} modulates what to look at in $h_1, \dots, h_T$ to compute s_i and sample y_i.



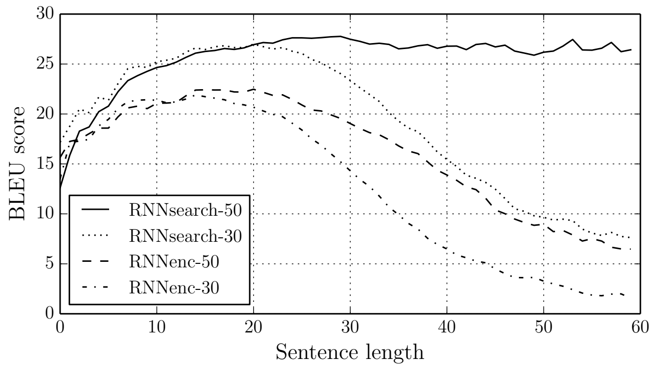
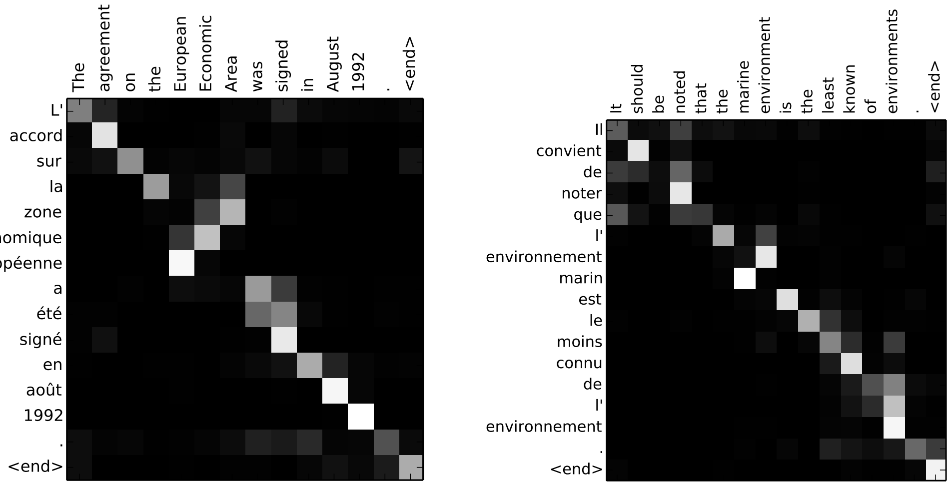


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

(Bahdanau et al., 2014)



(a) (b)

(Bahdanau et al., 2014)

References

D. Bahdanau, K. Cho, and Y. Bengio. **Neural machine translation by jointly learning to align and translate**. CoRR, abs/1409.0473, 2014.

K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. **Learning phrase representations using RNN encoder-decoder for statistical machine translation**. CoRR, abs/1406.1078, 2014.

A. Graves, G. Wayne, and I. Danihelka. **Neural turing machines**. CoRR, abs/1410.5401, 2014.

I. Sutskever, O. Vinyals, and Q. V. Le. **Sequence to sequence learning with neural networks**. In Neural Information Processing Systems (NIPS), pages 3104–3112, 2014.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. **Attention is all you need**. CoRR, abs/1706.03762, 2017.