Attention Models in Natural Language Processing

Tomasz Rembiasz, Juliusz Straszyński & Michał Maras

HUAWEI Warsaw Research Center

Phd Open

21 April 2022, University of Warsaw

Outline

HUAWEI

Outline

Attention Models in Machine Learning

- **Natural Languague Processing**
- Computer Vision
- Reinforcement Learning

Applications of attention in NLP

- Neural Machine Translation
- Question Answering (Q&A)
- Text Summarization
- Autocomplete
- Chatbot*
- Named Entity Recognition (NER)*
- Natural Language Understanding (NLU)*

* done at HUAWEI WRC

Chatbot

More task oriented

```
>>> I'm looking for a room in the city center.
<<< There are many options. Do you have a price range in mind?
>>> I'd prefer something cheap.
<<< I have two options for you. One is in the cheap price range and the other is in the expensive price rang
e. Which would you prefer?
>>> Cheap price range.
<<< I have the Acorn Guest House. Would you like me to book it for you?
>>> Where is it located?
<<< It is located at 154 chesterton road.
>>> Is it near the city center?
<<< Yes, it is in the city centre. Would you like me to book it for you?
>>> Ok, sure.
<<< How many people will be staying?
>>> Three.
<<< I have booked you a room at the Acorn Guest House. Your reference number is: 1ZY5VZQY. Is there anything
 else I can help you with?
```

More chit-chat style

```
>>> Do you have any hobby?
<<< I like to play the piano.
>>> What type of music do you play?
<<< I play the piano in a classical band.
>>> Sounds great! Will you play me something?
<<< Sure. What do you want to hear?
>>> Maybe some piece by Chopin, how about that?
<<< Sure.
```

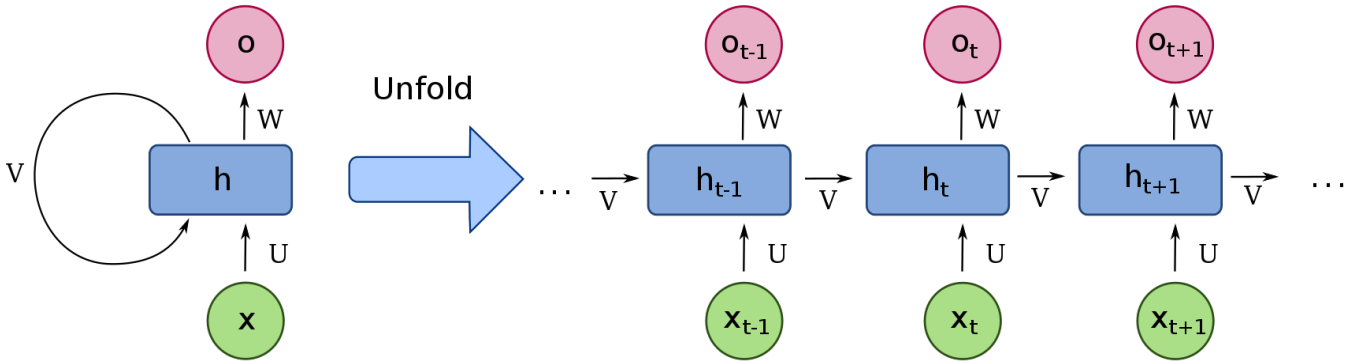
Conversation(s) with one of chatbots developed by our AI Team (Internship Project Summer 2021).

- Architecture: GPT-2
- Pretrained model: DialoGPT
- Dataset: custom mixture

HUAWEI

Outline

Recurrent Neural Networks (RNNs)



From Wikipedia

Selected applications:

- Machine translation
- Speech recognition
- Speech synthesis
- Grammar learning
- Music composition
- Handwriting recognition
- Protein homology detection
- Brain–computer interfaces

HUAWEI

Recurrent Neural Networks (RNNs)

words: one-hot vectors (of dim. v)

$$\mathbf{x}^{(t)} \in \mathbb{R}^v$$

word embeddings (of dimension n)

$$\mathbf{e}^{(t)} = \mathbf{E}\mathbf{x}^{(t)} \in \mathbb{R}^n$$

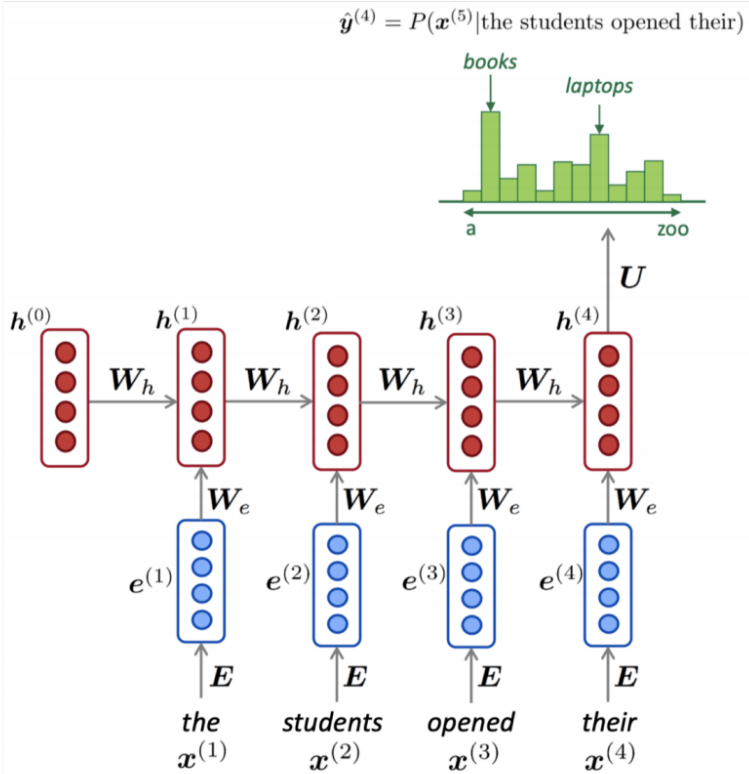
hidden states ($\mathbf{h}^{(0)}$ being initial)

$$\mathbf{h}^{(t)} = \sigma\left(\mathbf{W_h}\mathbf{h}^{(t-1)} + \mathbf{W_e}\mathbf{e}^{(t)} + \mathbf{b_1}\right)$$

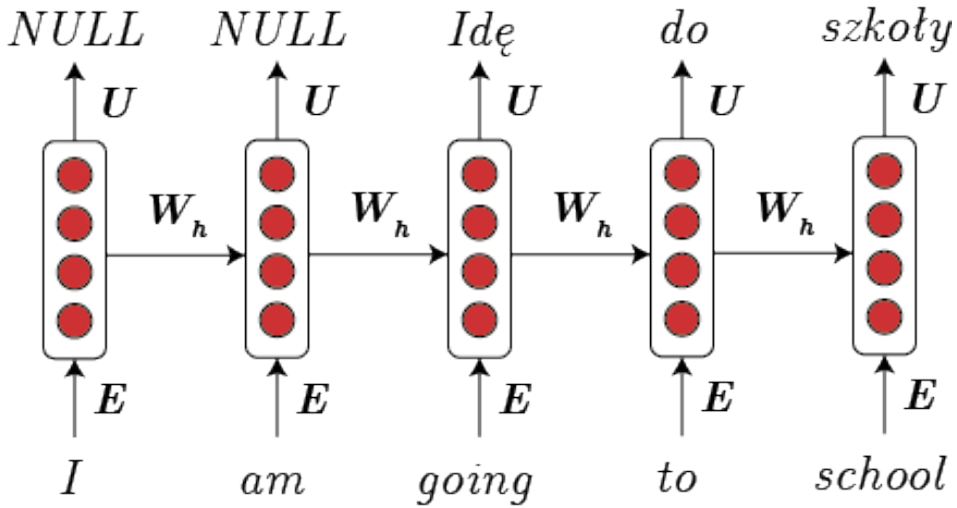
output distribution

$$\hat{\mathbf{y}}^{(t)} = \text{softmax}\left(\mathbf{U}\mathbf{h}^{(t)} + \mathbf{b_2}\right) \in \mathbb{R}^v$$

$$\mathbf{x}^{(t)} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$



Machine Translation with RNNs

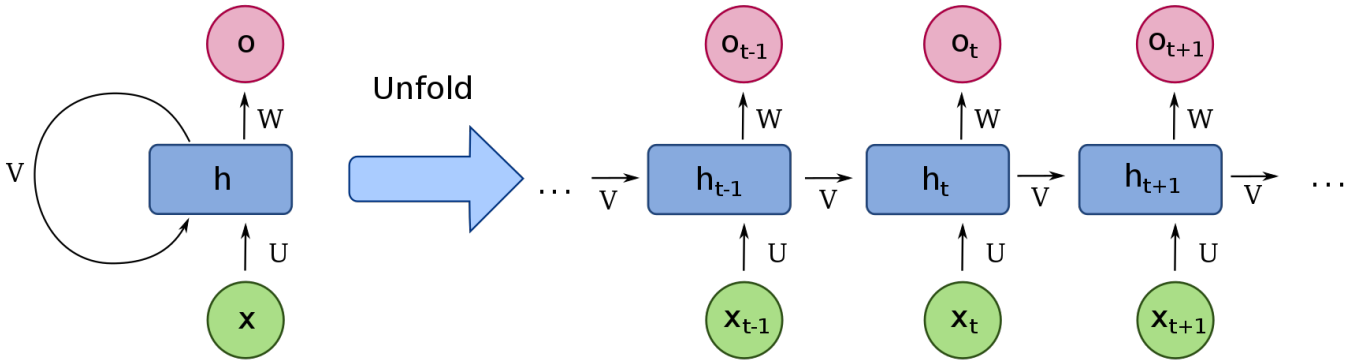


Sentences usually do not translate one-to-one.

I am going to school. $\rightarrow$ *Idę do szkoły.*

Alignment is the correspondence between particular words in the tranlsated sentence pair.

Drawbacks of RNNs

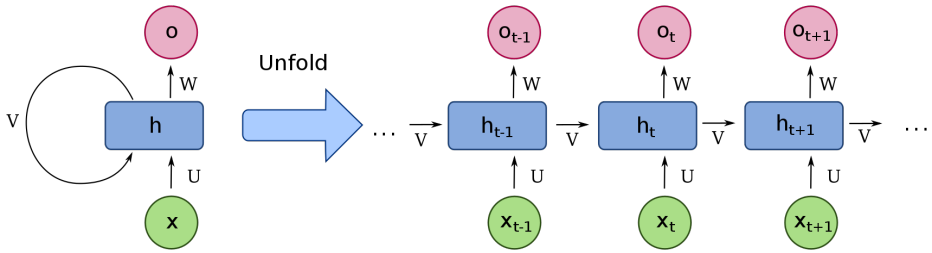


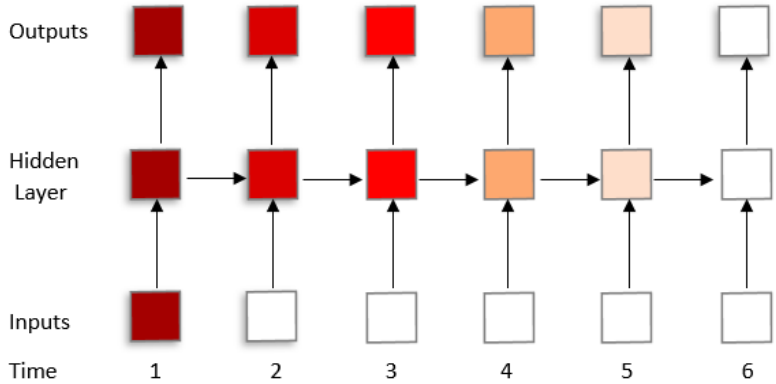
From Wikipedia

- Parallel computing difficult/impossible to implement
- Loss of information for long sequences
- Vanishing/exploding gradient problem

Vanishing gradient problem

Ich bin zu Hause, weil ich . . . [VERY LONG GERMAN SENTENCE] . . . bin.

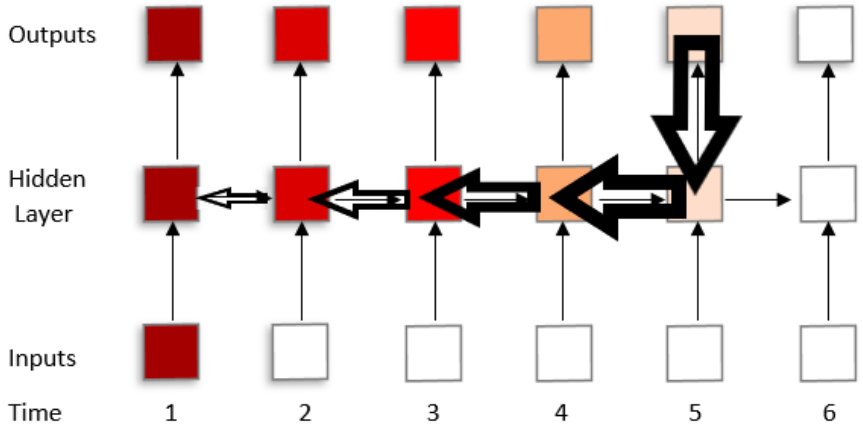




In RNNs, the hidden state gets constantly rewritten

$$\mathbf{h}^{(t)} = \sigma\left(\mathbf{V}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} + \mathbf{b}\right).$$

Vanishing gradient problem



$J^{(t)}(\theta)$ - loss function on step t.

$$\theta^{\text{new}} = \theta^{\text{old}} - \alpha \nabla_\theta J(\theta)$$

$$\frac{\partial J^{(t)}}{\partial \mathbf{h}^{(1)}} = \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}} \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}} \cdots \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} \frac{\partial J^{(t)}}{\partial \mathbf{h}^{(t)}}$$

Vanishing/exploding gradient problem

$$\mathbf{h}^{(t)} = \sigma\left(\mathbf{V}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} + \mathbf{b}\right)$$

assume $\sigma = \mathrm{Id}$, then

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} = \mathbf{V}$$

$$\frac{\partial J^{(t)}}{\partial \mathbf{h}^{(1)}} = \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}} \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}} \cdots \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} \frac{\partial J^{(t)}}{\partial \mathbf{h}^{(t)}} = \frac{\partial J^{(t)}}{\partial \mathbf{h}^{(t)}} \mathbf{V}^{t-1}$$

HUAWEI

Vanishing/exploding gradient problem

In basis of eigenvectors of $\mathbf{V}$, i.e.

$$\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_n}$$

with eigenvalues

$$\lambda_1, \lambda_2, \ldots, \lambda_n,$$

we can write

$$\frac{\partial J^{(t)}}{\partial \mathbf{h}^{(t)}} \mathbf{V}^{t-1} = \sum_{i=1}^{n} c_i \lambda_i^{t-1} \mathbf{q}_i.$$

For $t \gg 1$, if

$$\lambda_i < 1 \rightarrow \quad \text{vanishing gradient (memory needed!),}$$
$$\lambda_i > 1 \rightarrow \quad \text{exploding gradient (gradient clipping).}$$

Long Short-Term Memory (LSTM)

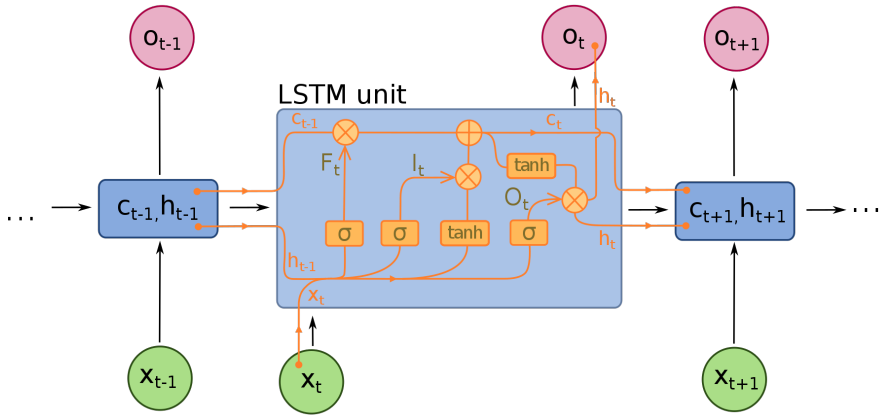
Hochreiter & Schmidhuber 1997

forget gate	$\mathbf{f}^{(t)} = \sigma\left(\mathbf{W_f h}^{(t-1)} + \mathbf{U_f x}^{(t)} + \mathbf{b_f}\right)$
input gate	$\mathbf{i}^{(t)} = \sigma\left(\mathbf{W_i h}^{(t-1)} + \mathbf{U_i x}^{(t)} + \mathbf{b_i}\right)$
output gate	$\mathbf{o}^{(t)} = \sigma\left(\mathbf{W_o h}^{(t-1)} + \mathbf{U_o x}^{(t)} + \mathbf{b_o}\right)$
new cell content	$\tilde{\mathbf{c}}^{(t)} = \tanh\left(\mathbf{W_c h}^{(t-1)} + \mathbf{U_c x}^{(t)} + \mathbf{b_c}\right)$
cell sate	$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}$
hidden state	$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh \mathbf{c}^{(t)}$

$$\mathbf{f}^{(t)}, \mathbf{i}^{(t)}, \mathbf{o}^{(t)}, \tilde{\mathbf{c}}^{(t)}, \mathbf{c}^{(t)}, \mathbf{h}^{(t)}, \mathbf{x}^{(t)} \in \mathbb{R}^n$$

HUAWEI

Long Short-Term Memory (LSTM)



Idea: Hochreiter & Schmidhuber 1997. Figure: Wikipedia

- cells store lon-term information
- read, write and erase options

$$\mathbf{f}^{(t)} = \sigma\left(\mathbf{W_f h}^{(t-1)} + \mathbf{U_f x}^{(t)} + \mathbf{b_f}\right)$$

$$\mathbf{i}^{(t)} = \sigma\left(\mathbf{W_i h}^{(t-1)} + \mathbf{U_i x}^{(t)} + \mathbf{b_i}\right)$$

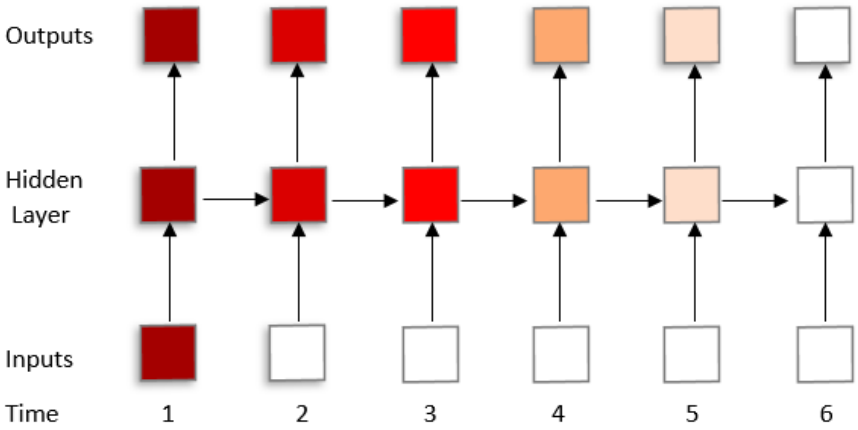
$$\mathbf{o}^{(t)} = \sigma\left(\mathbf{W_o h}^{(t-1)} + \mathbf{U_o x}^{(t)} + \mathbf{b_o}\right)$$

$$\tilde{\mathbf{c}}^{(t)} = \tanh\left(\mathbf{W_c h}^{(t-1)} + \mathbf{U_c x}^{(t)} + \mathbf{b_c}\right)$$

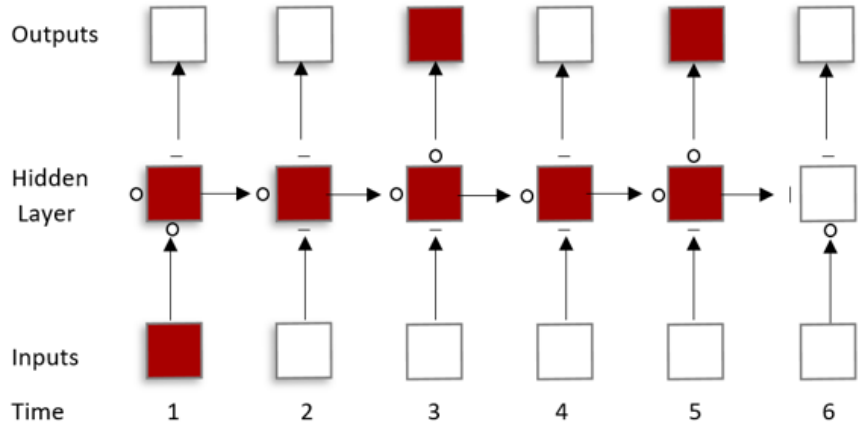
$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh \mathbf{c}^{(t)}$$

Vanishing gradient problem and LSTM



RNN without LSTM



RNN with LSTM

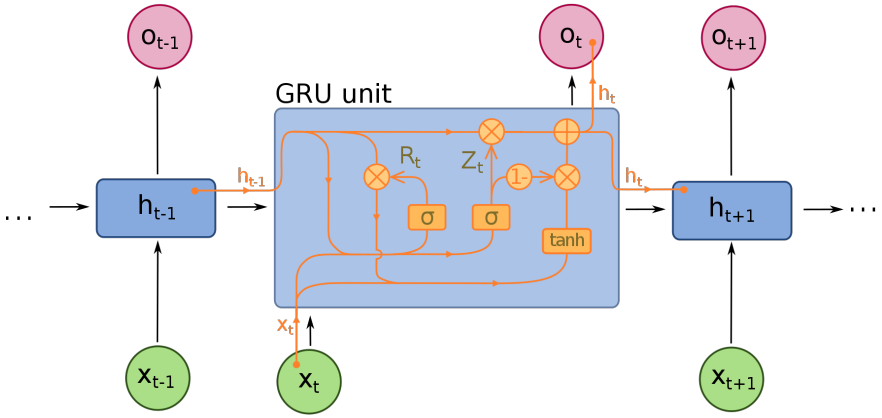
Gated Recurrent Units (GRUs)

Cho et al. 2014

update gate	$\mathbf{u}^{(t)} = \sigma\left(\mathbf{W_u h}^{(t-1)} + \mathbf{U_u x}^{(t)} + \mathbf{b_u}\right)$
reset gate	$\mathbf{r}^{(t)} = \sigma\left(\mathbf{W_r h}^{(t-1)} + \mathbf{U_r x}^{(t)} + \mathbf{b_r}\right)$
new hidden state content	$\tilde{\mathbf{h}}^{(t)} = \tanh\left(\mathbf{W_h}(\mathbf{r}^{(t)}\mathbf{h}^{(t-1)} + \mathbf{U_h x}^{(t)} + \mathbf{b_h}\right)$
hidden state	$\mathbf{h}^{(t)} = (1 - \mathbf{u}^{(t)}) \odot \mathbf{h}^{(t-1)} + \mathbf{u}^{(t)} \odot \tilde{\mathbf{h}}^{(t)}$

$$\mathbf{u}^{(t)}, \mathbf{r}^{(t)}, \tilde{\mathbf{h}}^{(t)}, \mathbf{h}^{(t)} \in \mathbb{R}^n$$

Gated Recurrent Units (GRUs)



Idea: Cho et al. 2014. Figure: Wikipedia

$$\mathbf{u}^{(t)} = \sigma\left(\mathbf{W_u}\mathbf{h}^{(t-1)} + \mathbf{U_u}\mathbf{x}^{(t)} + \mathbf{b_u}\right)$$

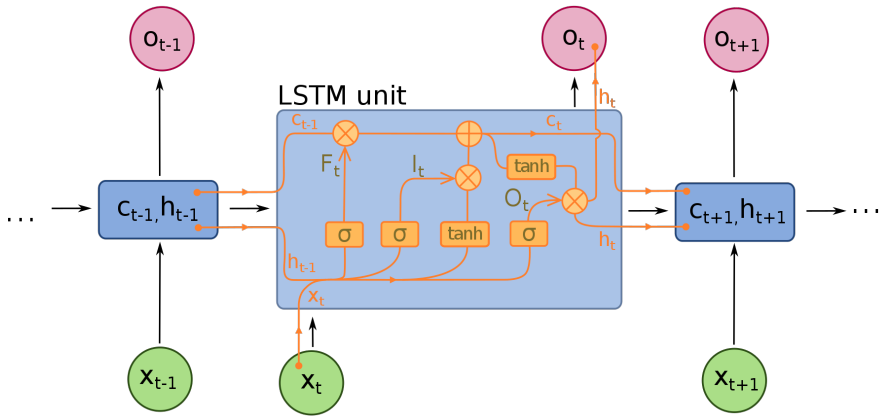
$$\mathbf{r}^{(t)} = \sigma\left(\mathbf{W_r}\mathbf{h}^{(t-1)} + \mathbf{U_r}\mathbf{x}^{(t)} + \mathbf{b_r}\right)$$

$$\tilde{\mathbf{h}}^{(t)} = \tanh\left(\mathbf{W_h}(\mathbf{r}^{(t)}\mathbf{h}^{(t-1)} + \mathbf{U_h}\mathbf{x}^{(t)} + \mathbf{b_h}\right)$$

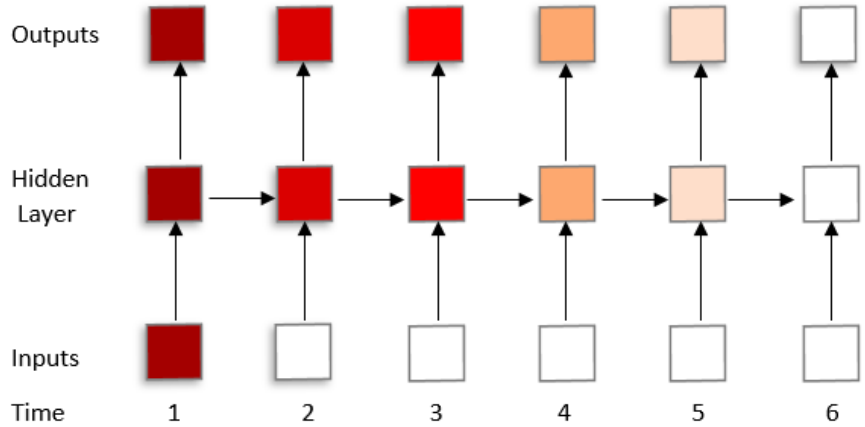
$$\mathbf{h}^{(t)} = (1 - \mathbf{u}^{(t)}) \odot \mathbf{h}^{(t-1)} + \mathbf{u}^{(t)} \odot \tilde{\mathbf{h}}^{(t)}$$

- No cell state
- Faster than LSTMs
- Fewer parameters

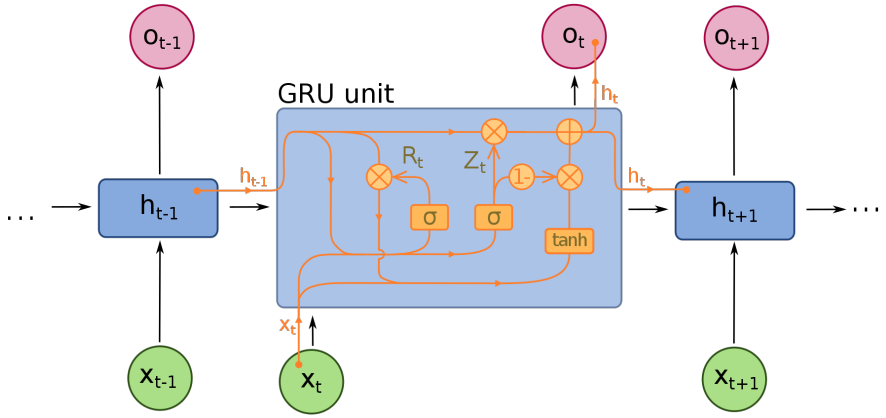
Vanishing gradient problem and LSTM & GRU gates



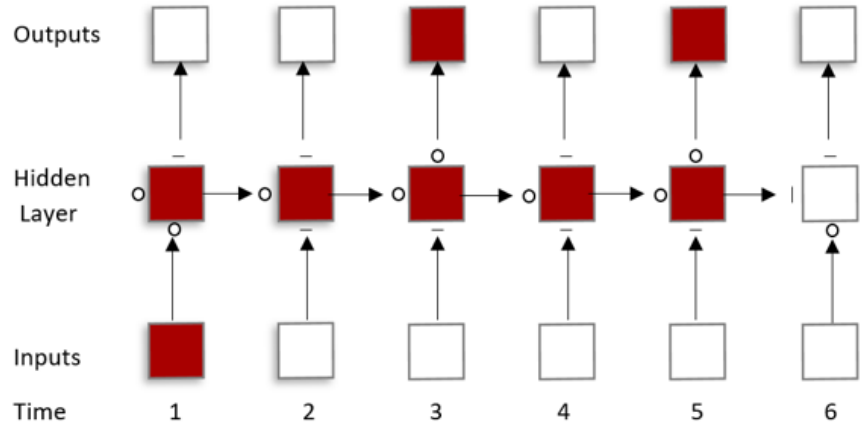
Idea: Hochreiter & Schmidhuber 1997. Figure: Wikipedia



Vanishing gradient problem



Idea: Cho et al. 2014. Figure: Wikipedia



RNN with LSTM

Outline

RNN encoder-decoder architecture

- Encoder:

input sentence $\mathbf{x} = (x_1, \ldots, x_{T_x})$ via

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

to context vector c:

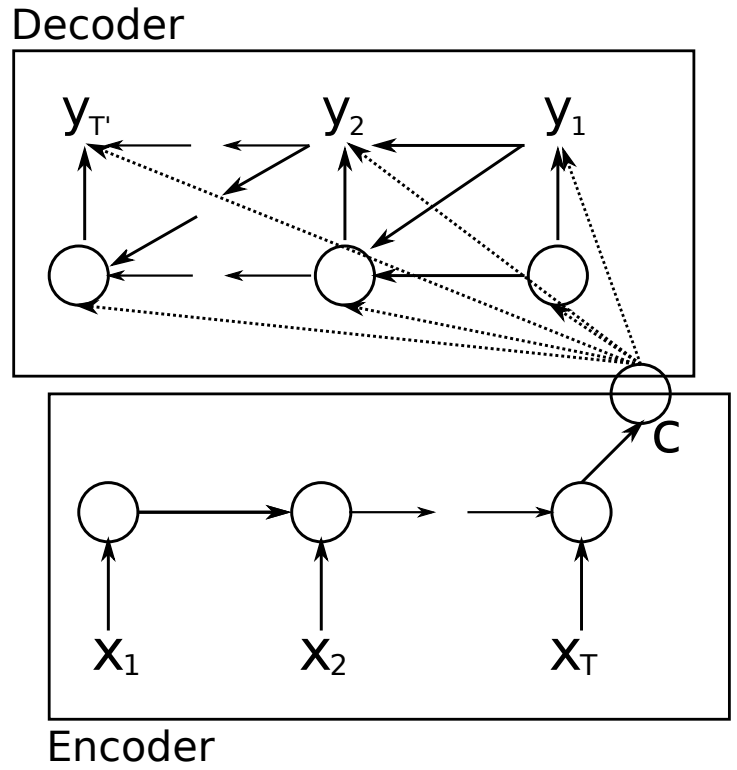
$$c = q\left(\{h_1, \ldots, h_{Tx}\}\right).$$

- Decoder:

next word y_t:

$$p(y_t \mid \{y_1, \ldots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

from previous ones and (fixed) c.



From Cho et al. 2014

RNN encoder-decoder architecture

Encoder converts the input sentence, a sequence of vectors $\mathbf{x} = (x_1, \ldots, x_{T_x})$, into a *context vector* c.

First, $h_t \in \mathbb{R}^n$, encoder hidden states at time t are calculated:

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

where f is a nonlinear function (eg. LSTM or GRU).

Next

$$c = q\left(\{h_1, \ldots, h_{T_x}\}\right),$$

where g is a nonlinear function, eg. $q\left(\{h_1, \ldots, h_{T_x}\}\right) = h_{T_x}$.

RNN encoder-decoder architecture

Decoder predicts the next word, y_t, given the (fixed) context vector, c, and all the previously predicted words $\{y_1, \ldots, y_{t-1}\}$:

$$p(\mathbf{y}) = \prod_{t=1}^{T_y} p(y_t \mid \{y_1, \ldots, y_{t-1}\}, c).$$

Each conditional probability is modeled as:

$$p(y_t \mid \{y_1, \ldots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c),$$

where s_t is a decoder hidden state:

$$s_t = f(s_{t-1}, y_{t-1}, c),$$

f and g are nonlinear functions.

Outline

Machine translation without attention

German sentence $\rightarrow$ (the whole meaning encoded in a single context vector if the sentence is too long some information gets) $\rightarrow$ English sentence

Attention

Without Attention

$$\text{German sentence} \rightarrow \begin{pmatrix} \vdots \\ \text{context} \\ \text{vector} \\ \vdots \end{pmatrix} \rightarrow \text{English sentence}$$

With Attention

Ich bin zu Hause, weil ich krank bin. $\rightarrow$ *I am home, because I am sick.*

- Sequence overload prevented.
- Model focuses on the likeliest word at each step.
- Uses a query (English word) to look in the right place to find the key (German word).

Attention, Alignment

- Decoder without attention (alignment):

$$p(y_t \mid \{y_1, \ldots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c),$$

where s_t is a decoder hidden state:

$$s_t = f(s_{t-1}, y_{t-1}, c),$$

and c is a fixed context vector.

- Decoder with attention (alignment):

$$p(y_i|\{y_1, \ldots, y_{i-1}\}, \mathbf{x}) = \tilde{g}(y_{i-1}, s_i, c_i),$$

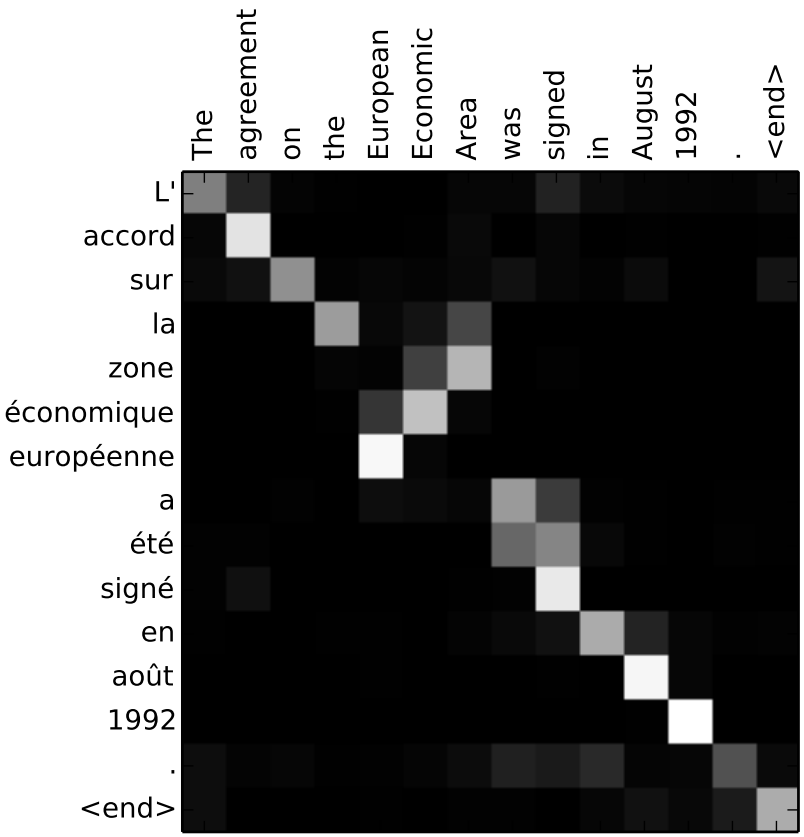
where s_i is a decoder hidden state:

$$s_i = \tilde{f}(s_{i-1}, y_{i-1}, c_i),$$

and c_i is a distinct context vector for each target word y_i.

Neural Machine Translation

$\alpha_{ij} \rightarrow j$ source (English)
$\downarrow i$ target (French)



From Bahdanau et al. 2015.

Context vector c_i

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

weights α_{ij} of annotations h_j

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

$$e_{ij} = a(s_{i-1}, h_j),$$

a alignment model (FF NN)
h_j encoder hidden state
s_{i-1} decoder hidden state

Dot-Product Attention

Dot-Product Attention

$$\text{Att}(Q, K, V) = \boxed{\text{softmax}\left(QK^T\right)}\boxed{V},$$

where

$Q \in \mathbb{R}^{n_{\text{out}} \times d_{\text{m}}}$ - queries (decoder)
$K \in \mathbb{R}^{n_{\text{in}} \times d_{\text{m}}}$ - keys (encoder)
$V \in \mathbb{R}^{n_{\text{in}} \times d_{\text{m}}}$ - values (encoder)
$n_{\text{in}}, n_{\text{out}}$ - sequence lengths,
d_{m} - embedding dimension

Additive Attention

Context vector c_i

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

weights α_{ij} of annotations h_j

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

$$e_{ij} = a(s_{i-1}, h_j),$$

a alignment model (FF NN)
h_j encoder hidden state
s_{i-1} decoder hidden state

Dot-Product Attention

$$\mathrm{Att}(Q, K, V) = \mathrm{softmax}\left(QK^T\right)V,$$

where the dot-product of the i-th query $Q_{i*} \in \mathbb{R}^{1\times d_{\mathsf{m}}}$ (i-th output token) with j-th key $K_{j*} \in \mathbb{R}^{1\times d_{\mathsf{m}}}$ (j-th input token) is

$$(QK^T)_{ij} = \sum_{l=1}^{d_{\mathsf{m}}} Q_{il}K_{jl}.$$

Function

$$\mathrm{softmax}(QK^T)_{ij} = \frac{\exp\left[\left(QK^T\right)_{ij}\right]}{\sum_{l=1}^{n_{\mathrm{in}}}\exp\left[\left(QK^T\right)_{il}\right]}$$

measures similarity between Q_{i*} and K_{j*}.

Dot-Product Attention

Hence, the Dot-Product Attention

$$\mathrm{Att}(Q, K, V) = \mathrm{softmax}\left(QK^T\right)V,$$

for the i-th query Q_{i*}, based on the most similar to it key K_{j*}, predominantly picks the j-th value V_{j*}.

In other words, for each query $Q_{i*} \in \mathbb{R}^{1\times d_\mathsf{m}}$, the Dot-Product Attention $\mathrm{Att}(Q, K, V)_{i*} \to \mathbb{R}^{1\times d_\mathsf{m}}$ is a weighted average of all values V_{j*}

$$\mathrm{Att}(Q, K, V)_{i*} = \sum_{j=1}^{n_{\mathrm{in}}} W_{ij} V_{j*},$$

with weights $W_{ij} = \mathrm{softmax}(QK^T)_{ij}$.

Additive Attention vs Dot-Product Attention

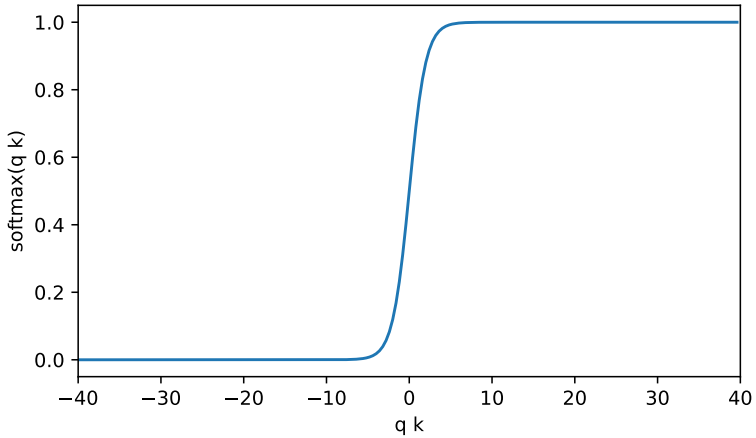
- Similar in theoretical complexity.
- Additive attention [AA] uses a feed-forward network with a single hidden layer.
- Dot-product (multiplicative) attention [DPA] can be faster and more space-efficient in practice when implemented with a highly optimized matrix multiplication code.
- For small d_{m}, both attentions perform similarly.
- For larger d_{m}, DPA gets slower than AA.
- To counteract it, Vaswani et al. 2017 introduced Scaled DPA .

Scaled Dot-Product Attention

Dot-Product Attention

$$\mathrm{Att}(Q, K, V) = \mathrm{softmax}\left(QK^T\right) V.$$

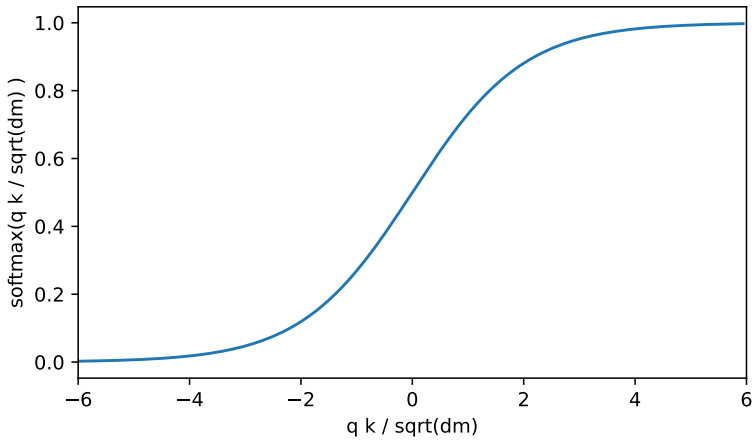
Each element $(QK^T)_{ij}$ has variance $\sim d_{\mathrm{m}}$.



Scaled Dot-Product Attention

$$\widetilde{\mathrm{Att}}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_{\mathrm{m}}}}\right) V.$$

Each $(QK^T/\sqrt{d_{\mathrm{m}}})_{ij}$ has variance ~ 1.



Variance in Dot-Product Attention

Matrix elements of the dot-product QK^T can get large for large d_{m}.
For arbitrarily fixed i and j, define $q = (q_1, \ldots, q_{d_{\text{m}}}) \equiv Q_{i*}$ and $k \equiv K_{j*}$.
Assume all q_α and k_β are independent random variables with mean $\mathrm{E}(q_\alpha) = \mathrm{E}(k_\beta) = 0$ and variance $\mathrm{Var}[q_\alpha] = \mathrm{Var}[k_\beta] = 1$. Then

$$\mathrm{E}[q \cdot k] = \mathrm{E}\left[\sum_{\alpha=1}^{d_{\text{m}}} q_\alpha k_\alpha\right] = 0,$$

because $\mathrm{Cov}[q_\alpha, k_\alpha] = 0$ and

$$\mathrm{Var}[q \cdot k] = \mathrm{E}[(q \cdot k)^2] = \mathrm{E}\left[\sum_{\alpha=1}^{d_{\text{m}}} \sum_{\beta=1}^{d_{\text{m}}} q_\alpha k_\alpha q_\beta k_\beta\right] = \sum_{\alpha=1}^{d_k} \mathrm{E}\left[q_\alpha^2\right] \mathrm{E}\left[k_\alpha^2\right] = d_{\text{m}},$$

because $\mathrm{Cov}[q_\alpha k_\alpha, q_\beta k_\beta] = \delta_{\alpha\beta}$ and $\mathrm{Cov}[q_\alpha^2, k_\alpha^2] = 0$.

HUAWEI


Thank
you
for
your
attention