Traducción de textos automatizada aplicando Deep Learning con Keras y Python

Pablo Vargas Ibarra

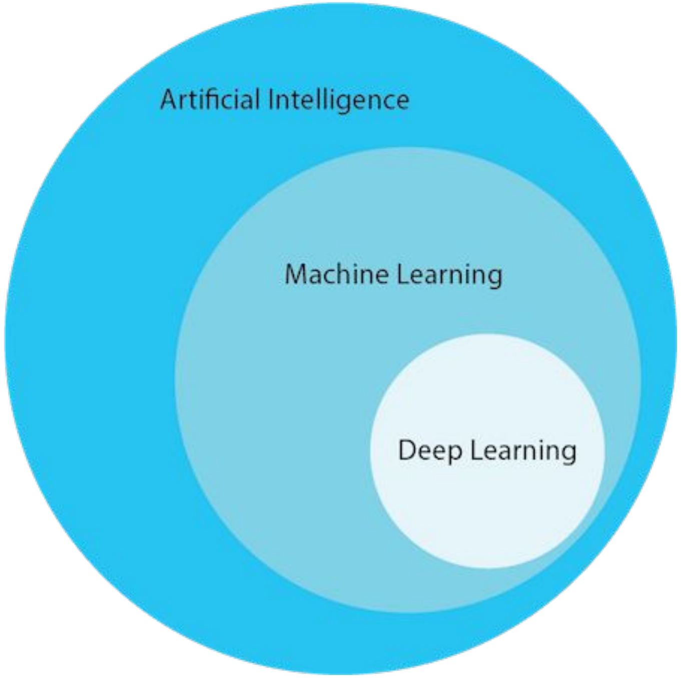
Málaga, 7 Octubre 2018

OUTLINE

INTRODUCTION

DL & Keras Introduction

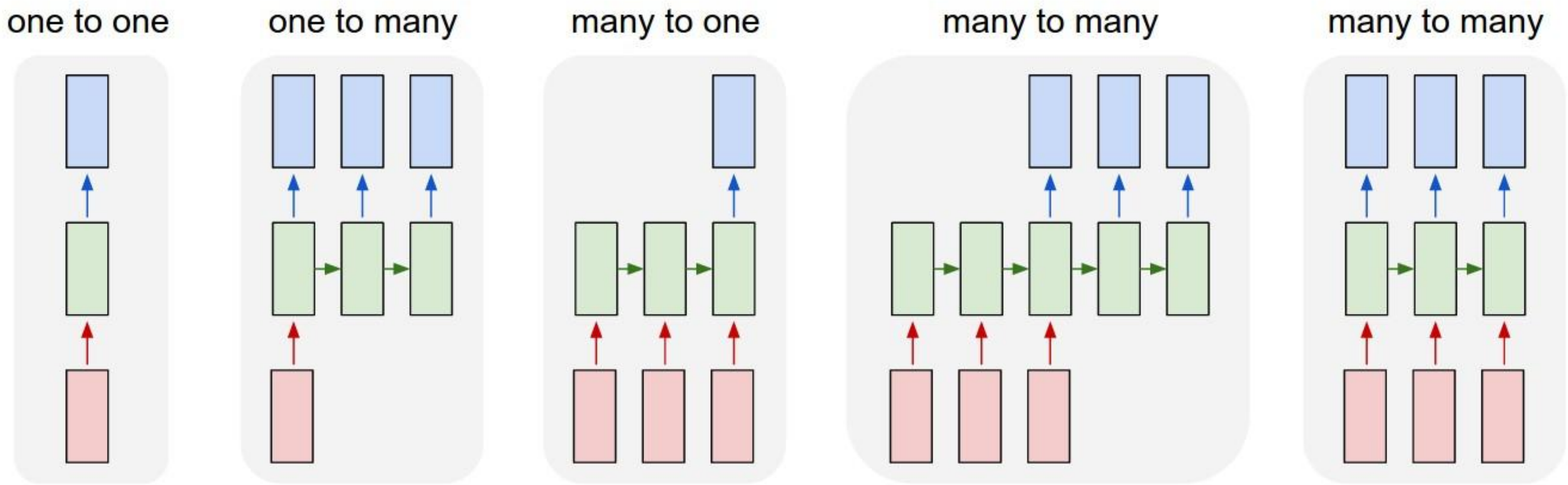
- Branch of machine learning
- Re-branded from neural networks
- State of the art results in speech recognition, computer vision, nlp and more…



- Deep Learning framework written in Python
- Open source, created in 2015
- Intuitive high-level API, easy to get started

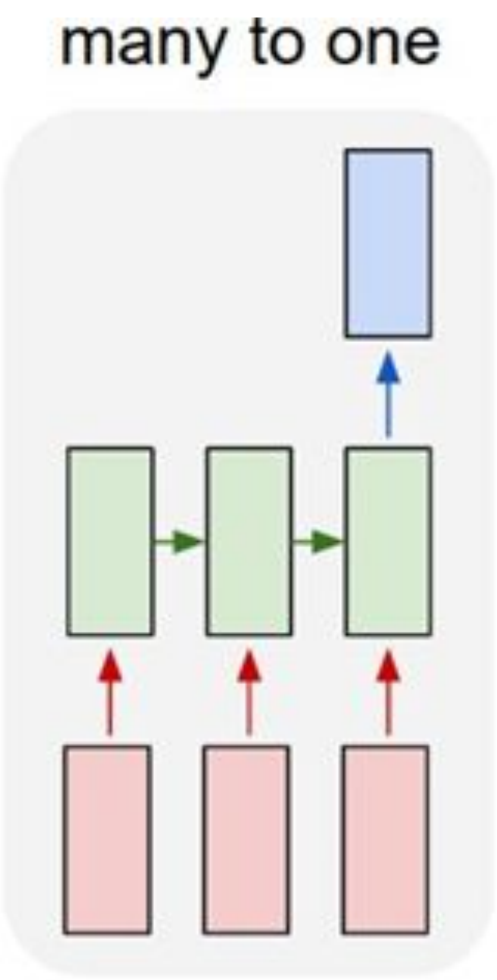
SEQUENCE MODELS

Sequence Models



Andrej Karpathy, The Unreasonable Effectiveness of Recurrent Neural Networks

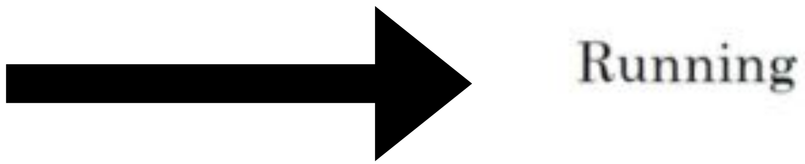
Sequence Models



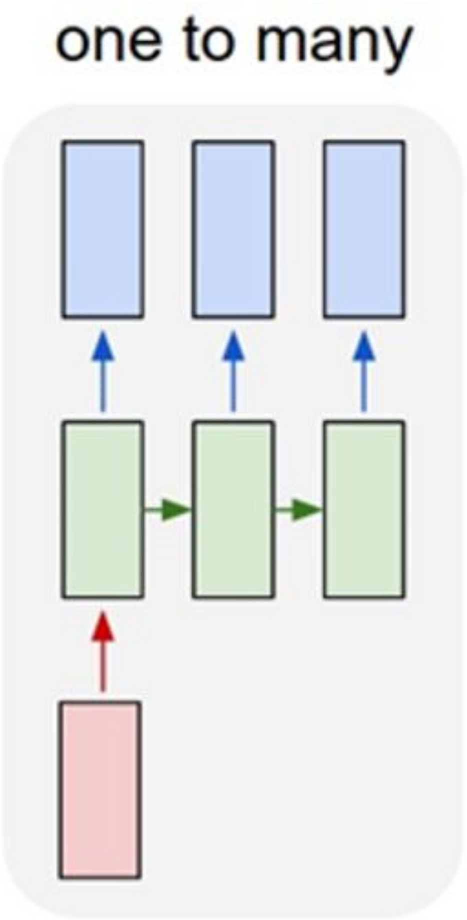
Time Series

1,2,3,4 ---> 5

Video Activity Recognition



Sequence Models



Language Model

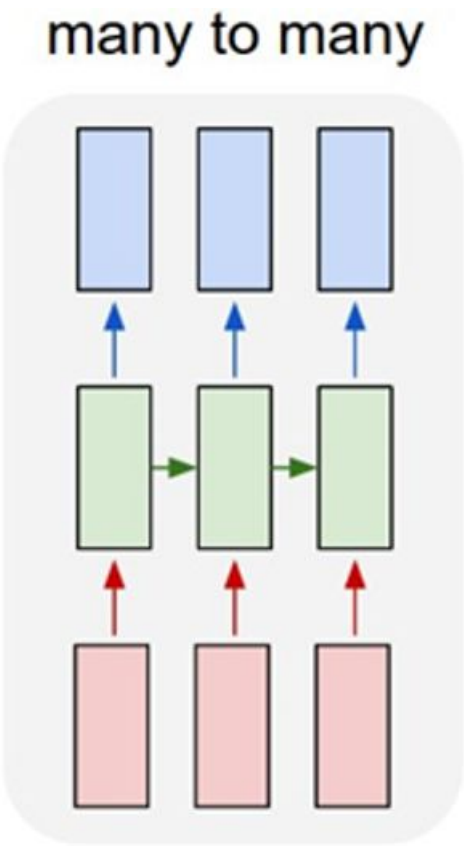
start

Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Image Captioning

A person riding a motorcycle on a dirt road.

Sequence Models

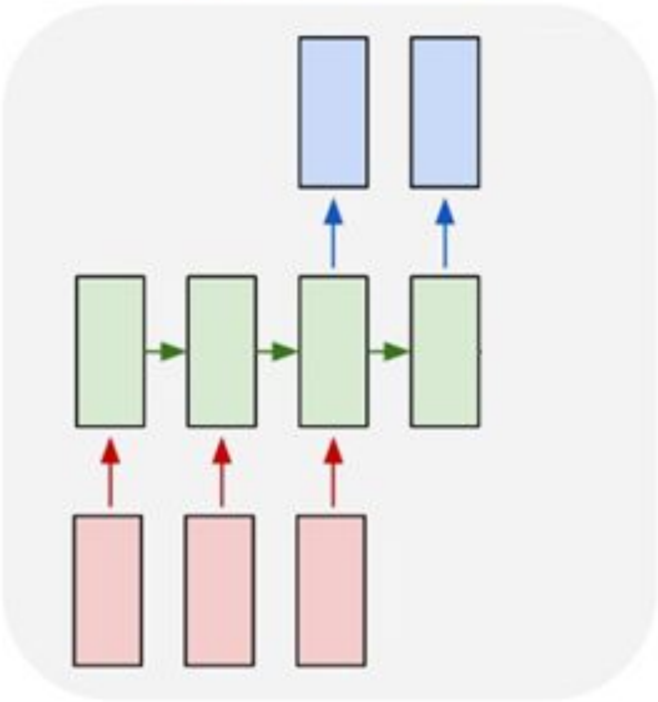


Name-entity recognition

Yesterday, Harry Potter met Hermione Granger. → Yesterday, Harry Potter met Hermione Granger.

Sequence Models

many to many

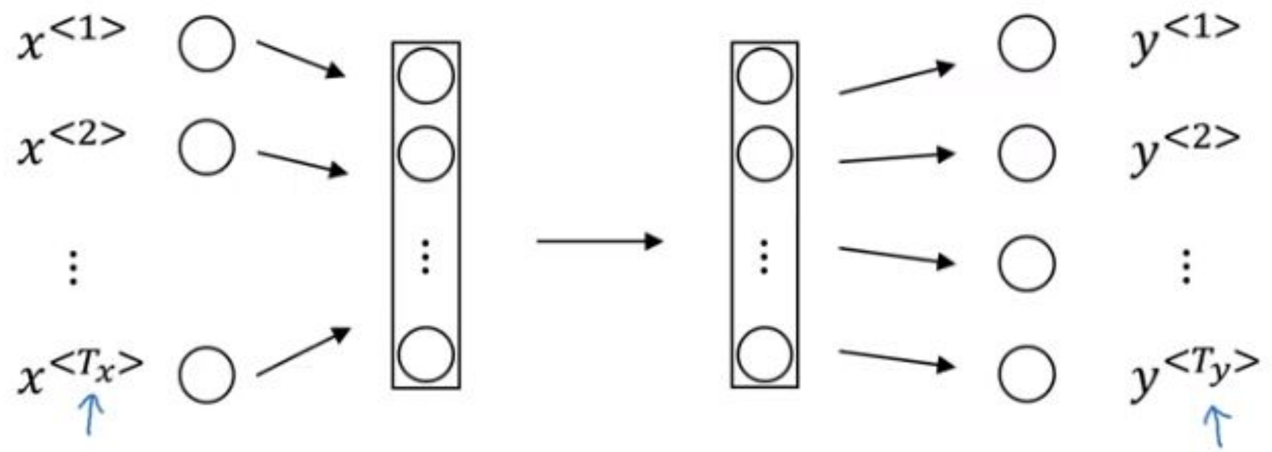


Machine Translation

Voulez-vous chanter avec moi? → Do you want to sing with me?

Sequence Models

Why not a standard network?



Problems:

- Inputs, outputs can be different lengths in different examples.
- Doesn't share features learned across different positions of text.

Andrew Ng

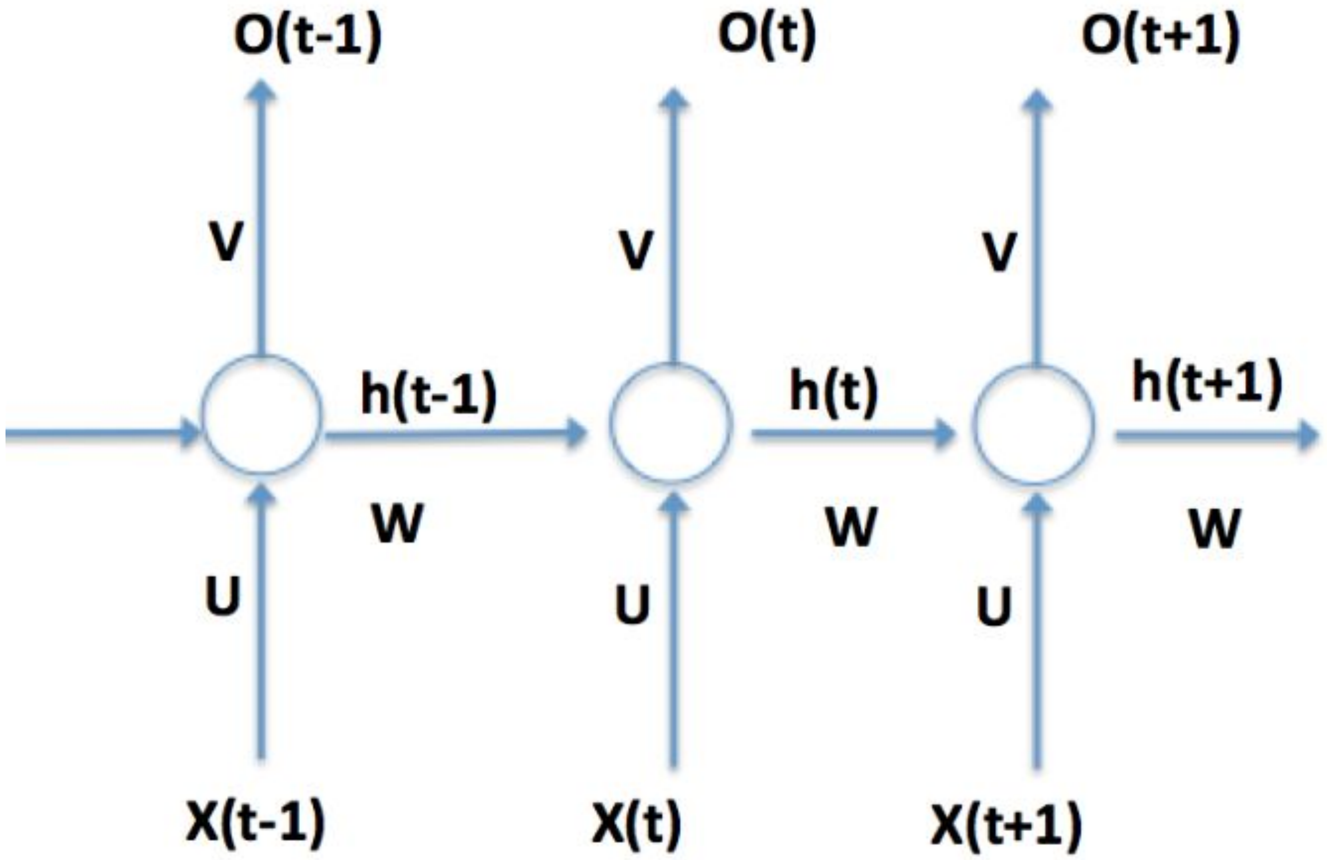
Sequence Models

Me parecen interesantes las matemáticas.

Las matemáticas me parecen interesantes.

RECURRENT NETWORKS

Recurrent Neural Networks

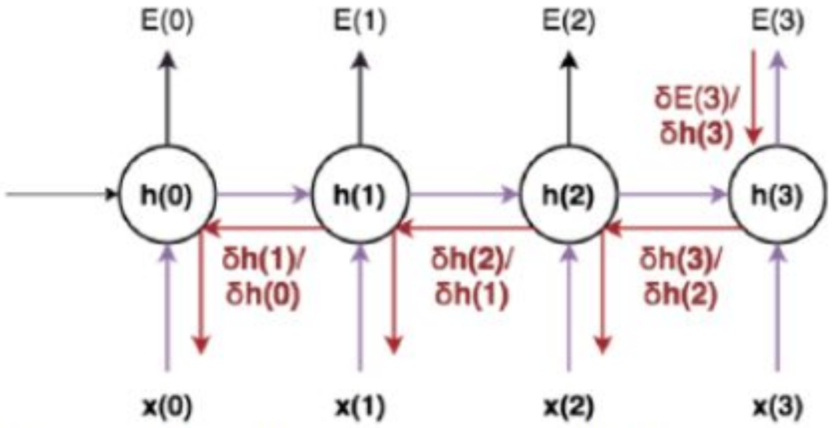


$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\ \hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(\boldsymbol{o}^{(t)}) \end{aligned}$$

Recurrent Neural Networks

Backpropagation through time: Just like we sum up the errors at output, we sum up the gradients at each time step

$$\frac{\partial E_3}{\partial W} = \sum_{h=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \frac{\partial h_3}{\partial h_k} \frac{\partial h_k}{\partial W}$$

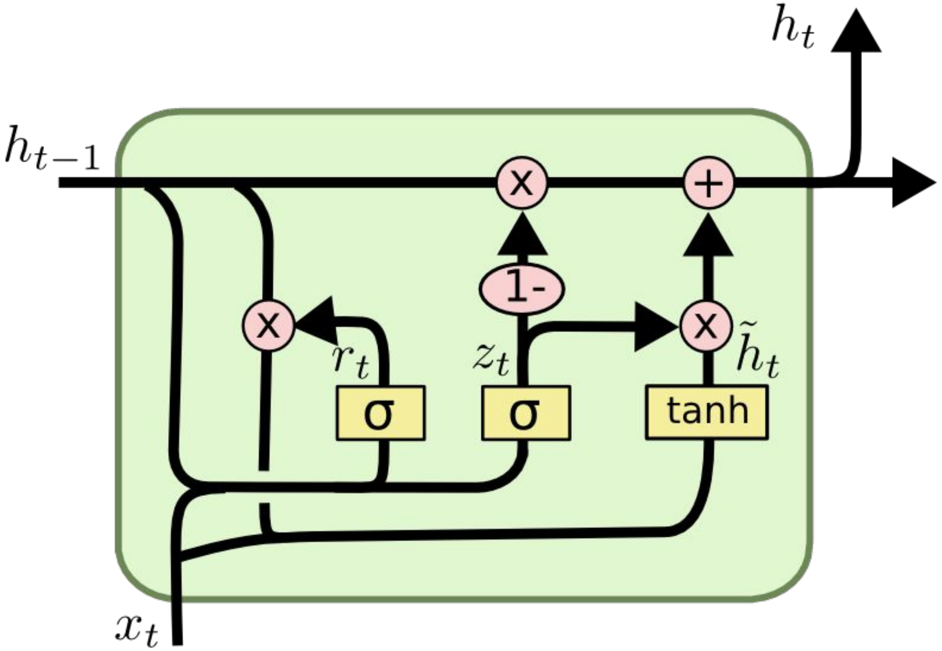


Example back-prop in time with 3 time-steps

Recurrent Neural Networks

Gated Recurrent Units (**GRU**), Kyunghyun Cho et al. (2014)

Long Short-Term Memory (**LSTM**), Hochreiter & Schmidhuber (1997)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

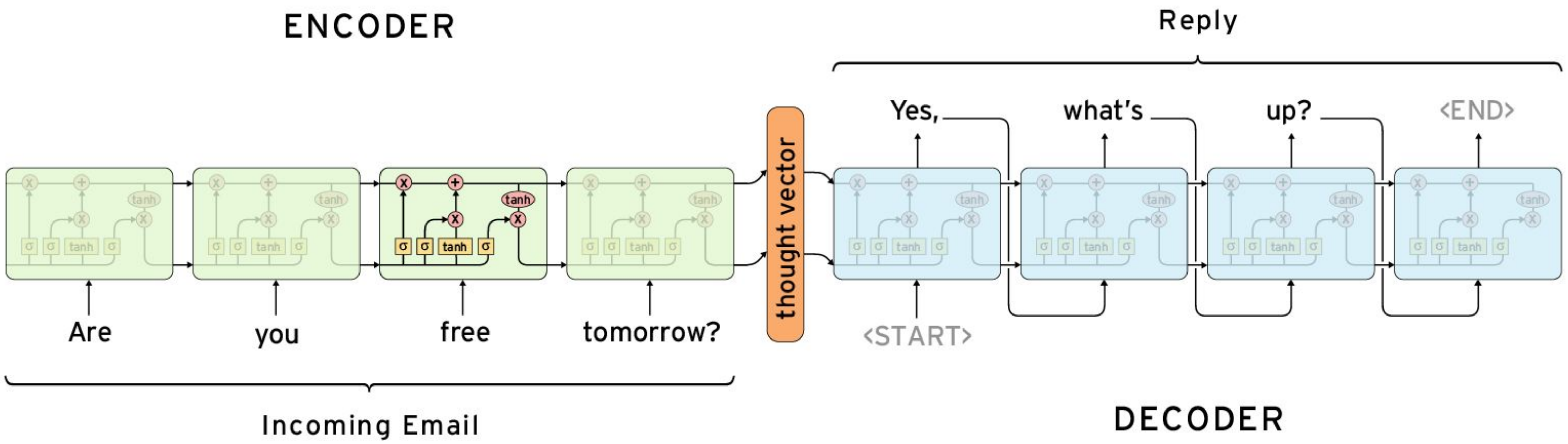
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Understanding LSTM Networks http://colah.github.io/posts/2015-08-Understanding-LSTMs/

NEURAL MACHINE TRANSLATION

Neural Machine Translation

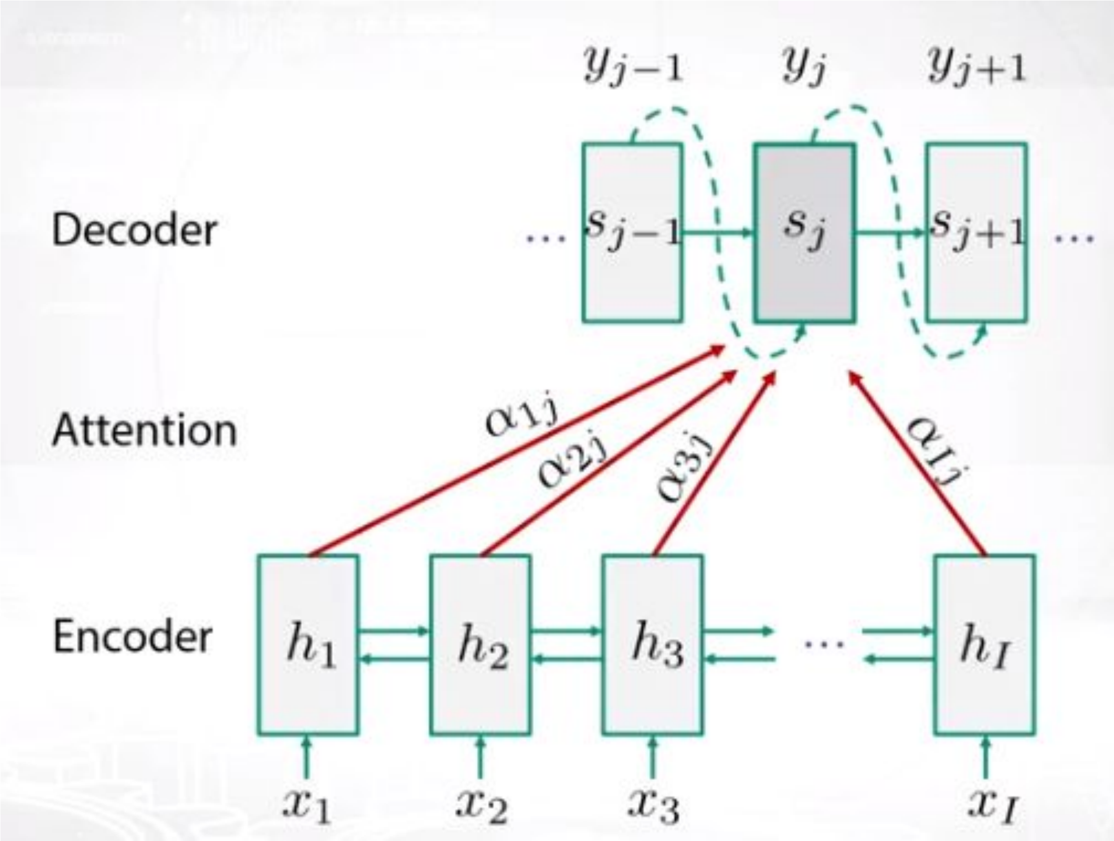
seq2seq: RNN as an encoder-decoder structure



Ilya Sutskever, Oriol Vinyals, Quoc V. Le Sequente to Sequence Learning with Neural Network

Neural Machine Translation

Attention Mechanism



- **Additive attention:**

$$sim(h_i, s_j) = w^T \tanh(W_h h_i + W_s s_j)$$

- **Multiplicative attention:**

$$sim(h_i, s_j) = h_i^T W s_j$$

- **Dot product also works:**

$$sim(h_i, s_j) = h_i^T s_j$$

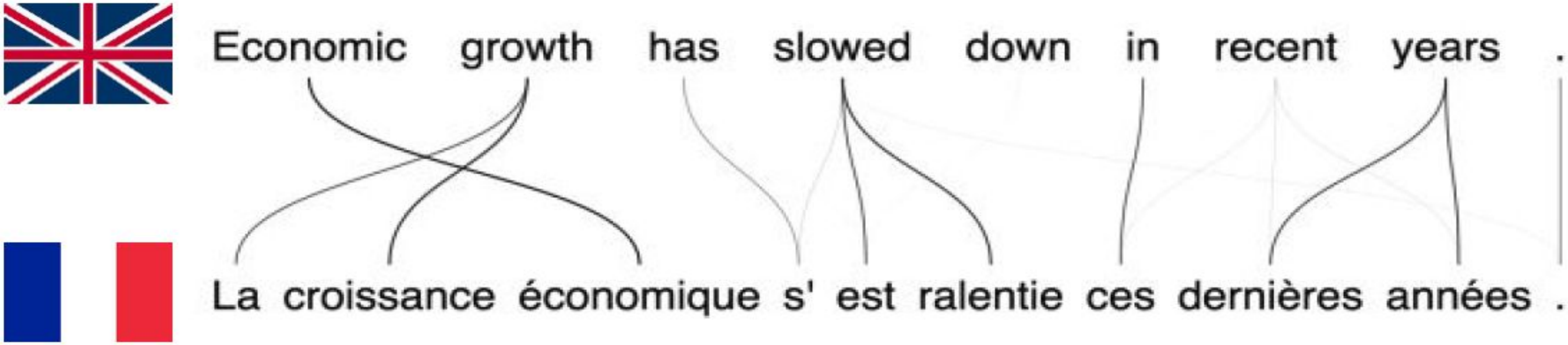
D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. (2014)

HSE (2018). Natual Language Processing. Coursera.

Neural Machine Translation

Attention Mechanism

The model automatically finds the correspondence structure between two languages (alignment).

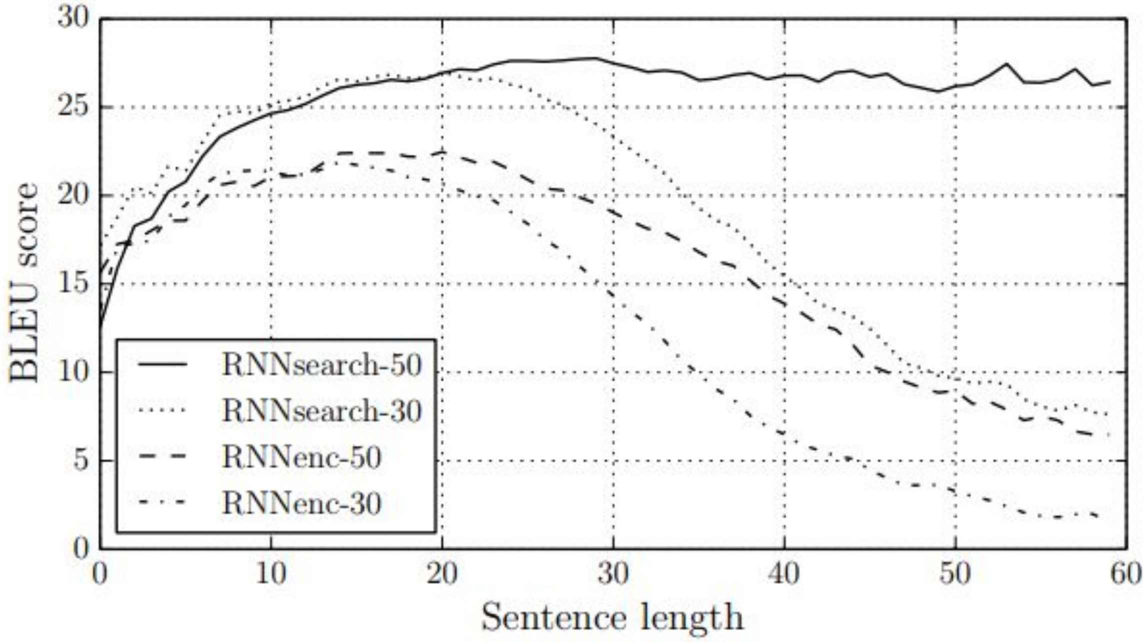


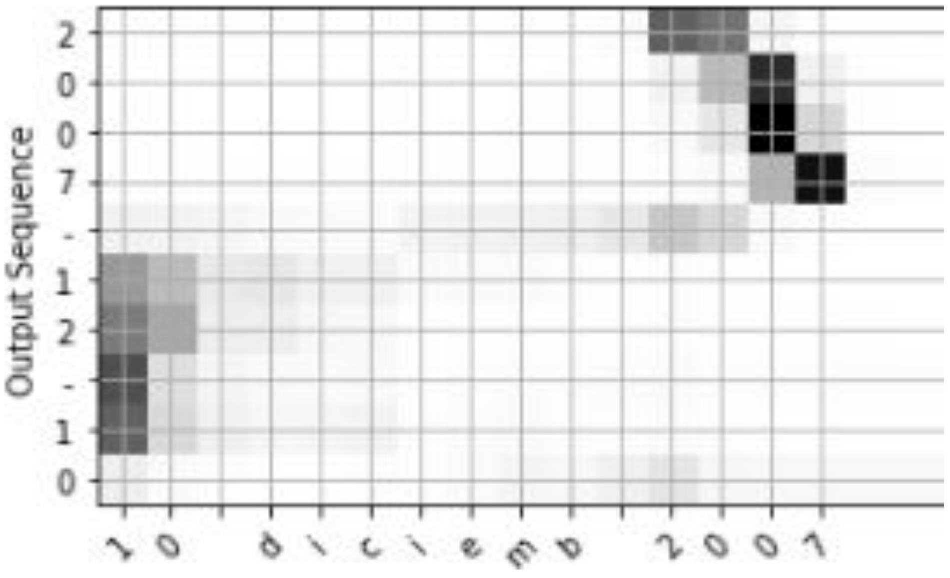
(Edge thicknesses represent the attention weights found by the attention model)

Neural Machine Translation

Attention Mechanism

Data : ACL WMT ’14
RNN Enc-Dec vs RNN-Search





D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. (2014)

Example of problem

Example of problem

9 mayo 1998 → 1998-05-09

10.09.70 → 1970-09-10

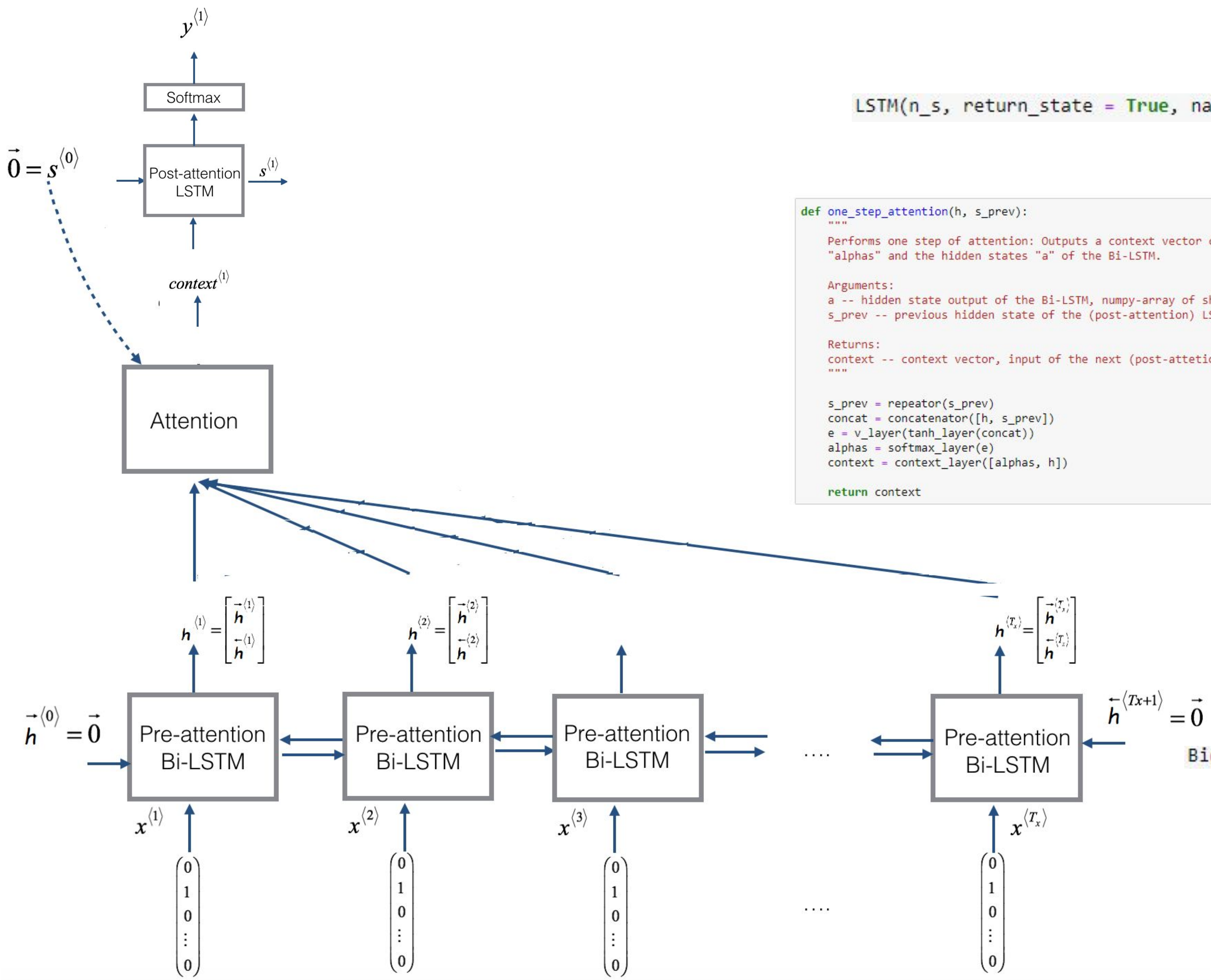
december 24 of 1978 → 1978-12-24

```
x.shape: (28, 38)
```

```
y.shape: (10, 11)
```

```
In [5]: human_vocab
Out[5]: {'9': 0,
         ' ': 1,
         'm': 2,
         'a': 3,
         'y': 4,
         'o': 5,
         '1': 6,
         '8': 7,
         '0': 8,
         '.': 9,
         '7': 10,
         '4': 11,
         '/': 12,

        'w': 35,
        '#': 36,
        'UNK': 37}
```

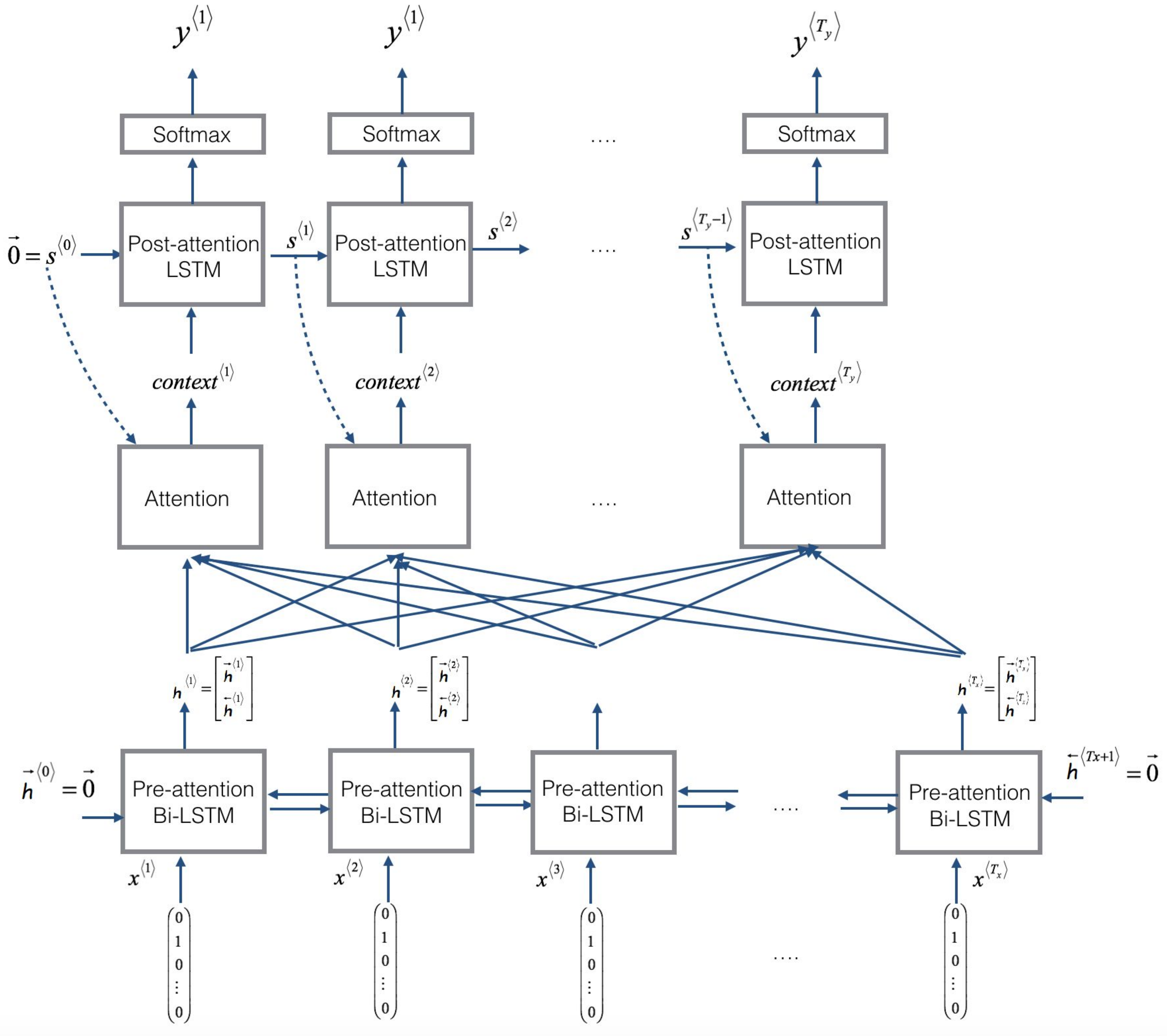
```
In [6]: machine_vocab
Out[6]: {'1': 0,
         '9': 1,
         '8': 2,
         '-': 3,
         '0': 4,
         '5': 5,
         '7': 6,
         '4': 7,
         '2': 8,
         '6': 9,
         '3': 10}
```




```
LSTM(n_s, return_state = True, name='post-attention-LSTM')
```

```
def one_step_attention(h, s_prev):
    """
    Performs one step of attention: Outputs a context vector computed as a dot product of the attention weights
    "alphas" and the hidden states "a" of the Bi-LSTM.

    Arguments:
    a -- hidden state output of the Bi-LSTM, numpy-array of shape (m, Tx, 2*n_a)
    s_prev -- previous hidden state of the (post-attention) LSTM, numpy-array of shape (m, n_s)

    Returns:
    context -- context vector, input of the next (post-attetion) LSTM cell
    """

    s_prev = repeator(s_prev)
    concat = concatenator([h, s_prev])
    e = v_layer(tanh_layer(concat))
    alphas = softmax_layer(e)
    context = context_layer([alphas, h])

    return context
```

```
Bidirectional(LSTM(n_h, return_sequences=True))
```



$y^{\langle 1 \rangle}$
$y^{\langle 1 \rangle}$
$y^{\langle T_y \rangle}$
Softmax
Softmax
....
Softmax
$\vec{0} = s^{\langle 0 \rangle}$
Post-attention LSTM
$s^{\langle 1 \rangle}$
Post-attention LSTM
$s^{\langle 2 \rangle}$
....
$s^{\langle T_y - 1 \rangle}$
Post-attention LSTM
$context^{\langle 1 \rangle}$
$context^{\langle 2 \rangle}$
$context^{\langle T_y \rangle}$
Attention
Attention
....
Attention
$h^{\langle 1 \rangle} = \begin{bmatrix} \vec{h}^{\langle 1 \rangle} \\ \overleftarrow{h}^{\langle 1 \rangle} \end{bmatrix}$
$h^{\langle 2 \rangle} = \begin{bmatrix} \vec{h}^{\langle 2 \rangle} \\ \overleftarrow{h}^{\langle 2 \rangle} \end{bmatrix}$
$h^{\langle T_x \rangle} = \begin{bmatrix} \vec{h}^{\langle T_x \rangle} \\ \overleftarrow{h}^{\langle T_x \rangle} \end{bmatrix}$
$\vec{h}^{\langle 0 \rangle} = \vec{0}$
Pre-attention Bi-LSTM
Pre-attention Bi-LSTM
Pre-attention Bi-LSTM
....
Pre-attention Bi-LSTM
$\overleftarrow{h}^{\langle Tx+1 \rangle} = \vec{0}$
$x^{\langle 1 \rangle}$
$x^{\langle 2 \rangle}$
$x^{\langle 3 \rangle}$
$x^{\langle T_x \rangle}$
....

BIBLIOGRAPHY

SOME BIBLIOGRAPHY

I. Goodfellow, Y. Bengio, & A. Courville (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org

A. Ng (2017). Deep Learning Specialization. Coursera.
https://www.coursera.org/specializations/deep-learning

D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. (2014) https://arxiv.org/abs/1409.0473

HSE (2018). Natual Language Processing. Coursera. https://www.coursera.org/learn/language-processing

Ilya Sutskever, Oriol Vinyals, Quoc V. Le Sequente to Sequence Learning with Neural Network.
https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

Andrej Karpathy, The Unreasonable Effectiveness of Recurrent Neural Networks
http://karpathy.github.io/2015/05/21/rnn-effectiveness/