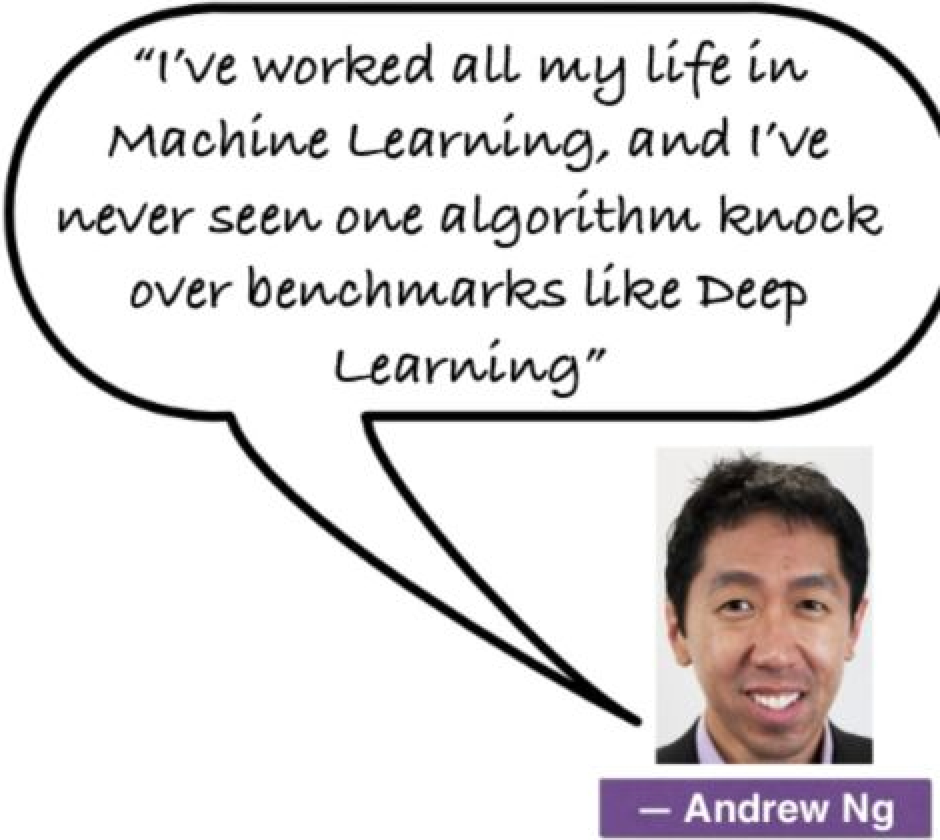
Deep learning for NLP

Models of meaning

Why Deep learning?



Let's start with word vectors

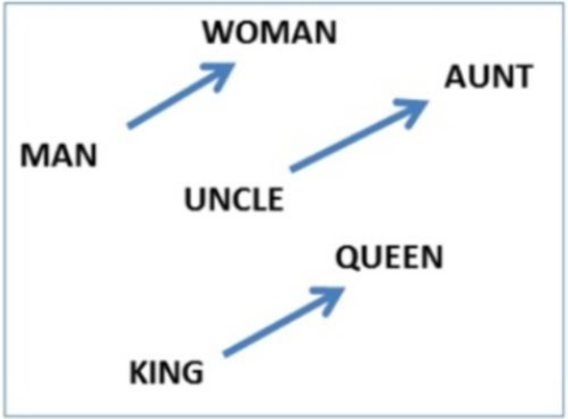
1. The first try: one-hot encoding

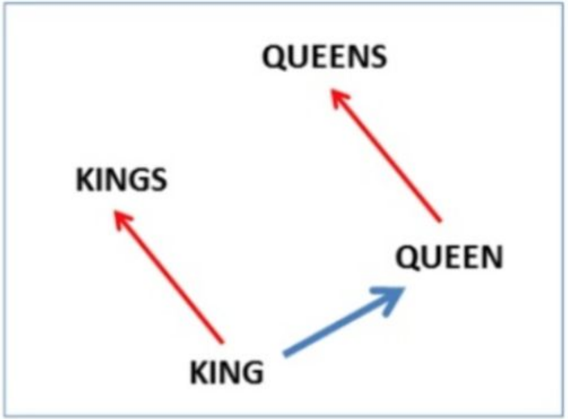
```
V = {zebra, horse, school, summer}

 v(zebra) = [1, 0, 0, 0]
 v(horse) = [0, 1, 0, 0]
v(school) = [0, 0, 1, 0]
v(summer) = [0, 0, 0, 1]
```

The second try: word2vec and GloVe

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

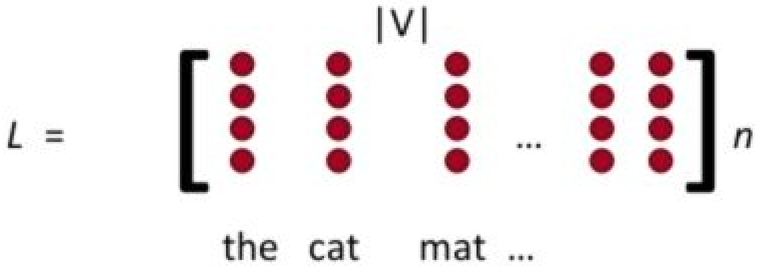
vec(“man”) – vec(“king”) + vec(“woman”) = vec(“queen”)

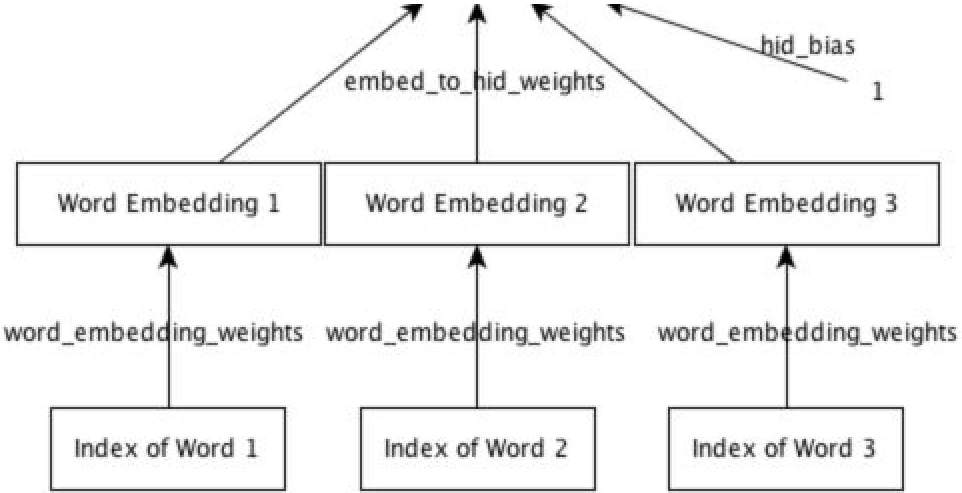




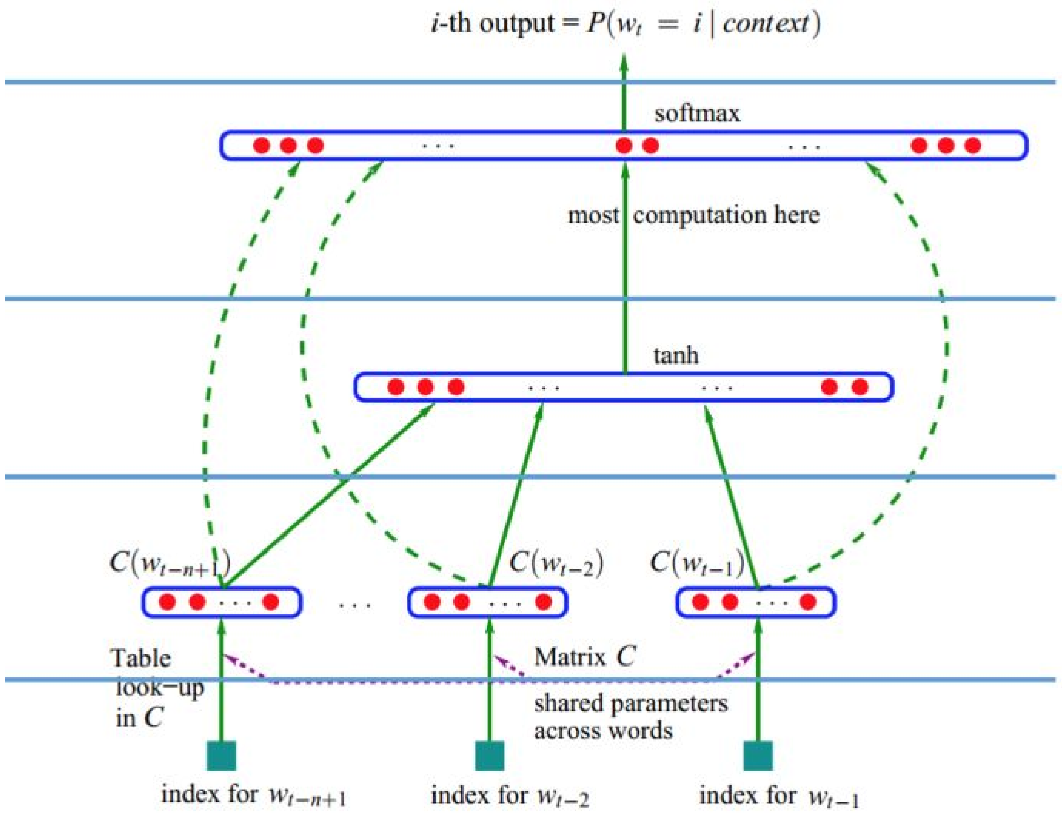
Embedding layer

Embedding matrix:





Feedforward Neural Networks for classification



Training Neural Networks

Indexes of words

↓

Embedding layer

↓

Neural Network

↓

Output (Softmax)

↓

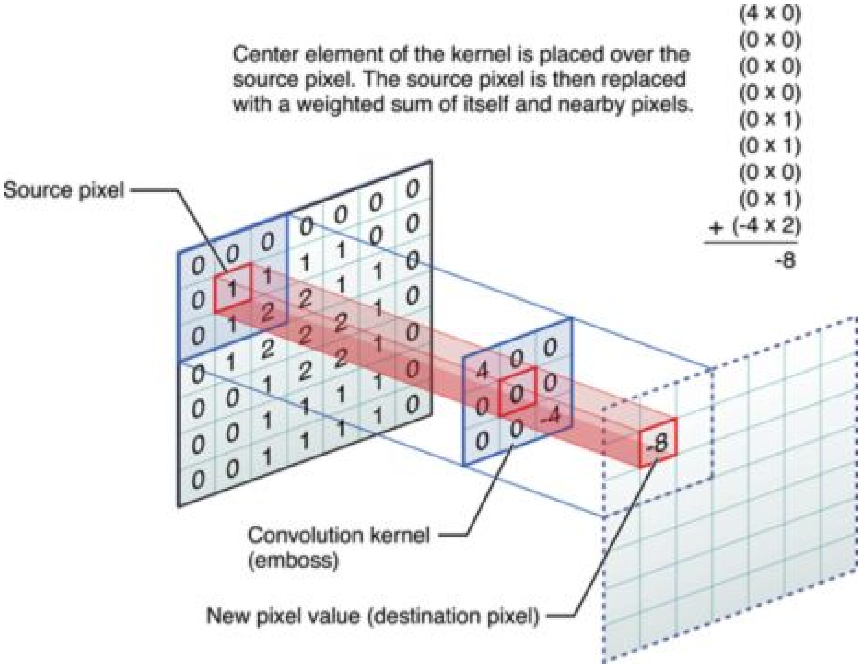
Loss function (J)

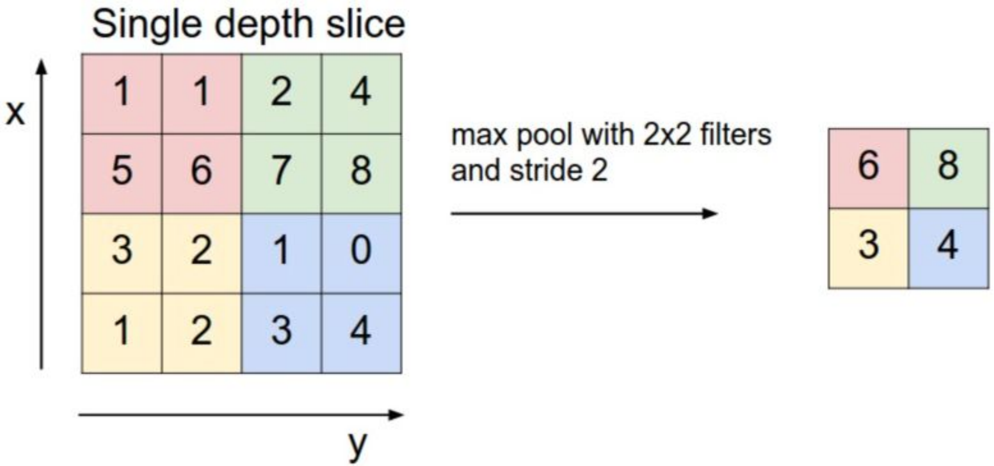
↓

SGD on mini-batches

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

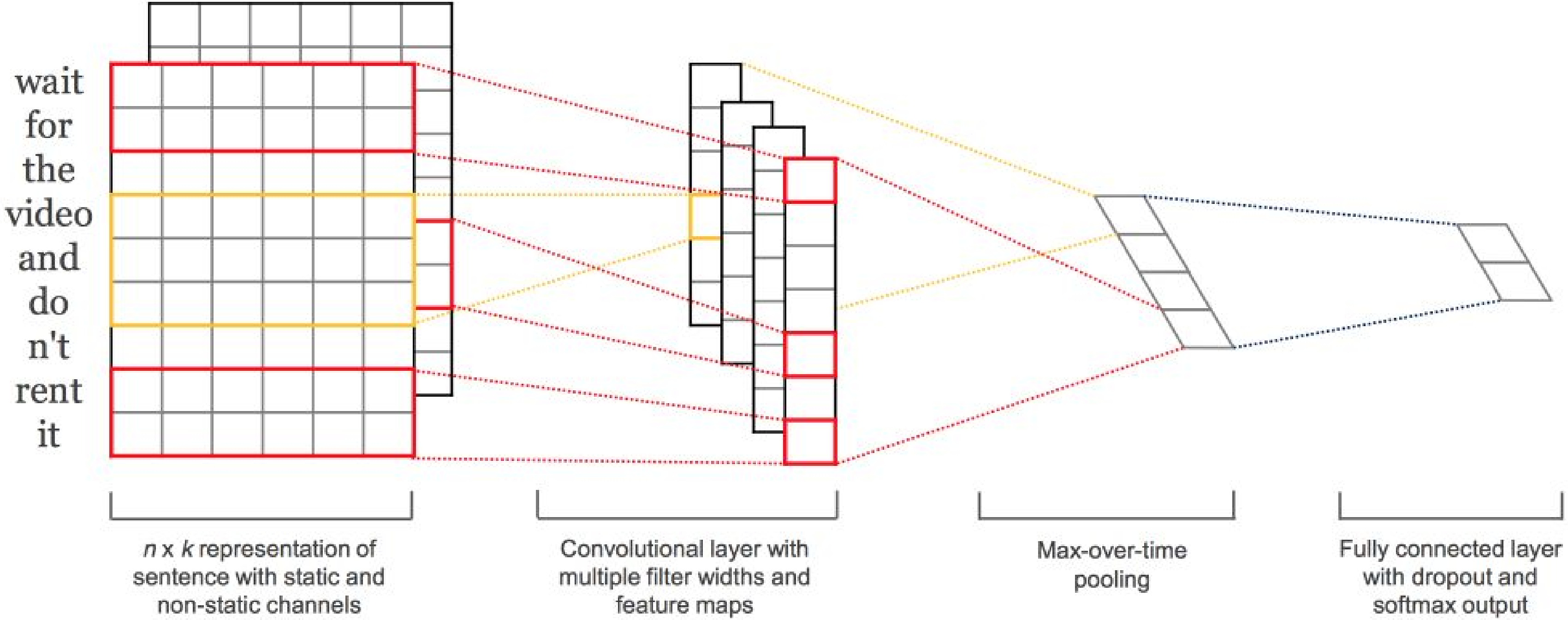
Convolution and pooling operations





1D/2D Convolutional neural networks

Text is not a picture! We have to convolve over words.



http://arxiv.org/pdf/1408.5882v2.pdf

Dynamic Convolutional Neural Network

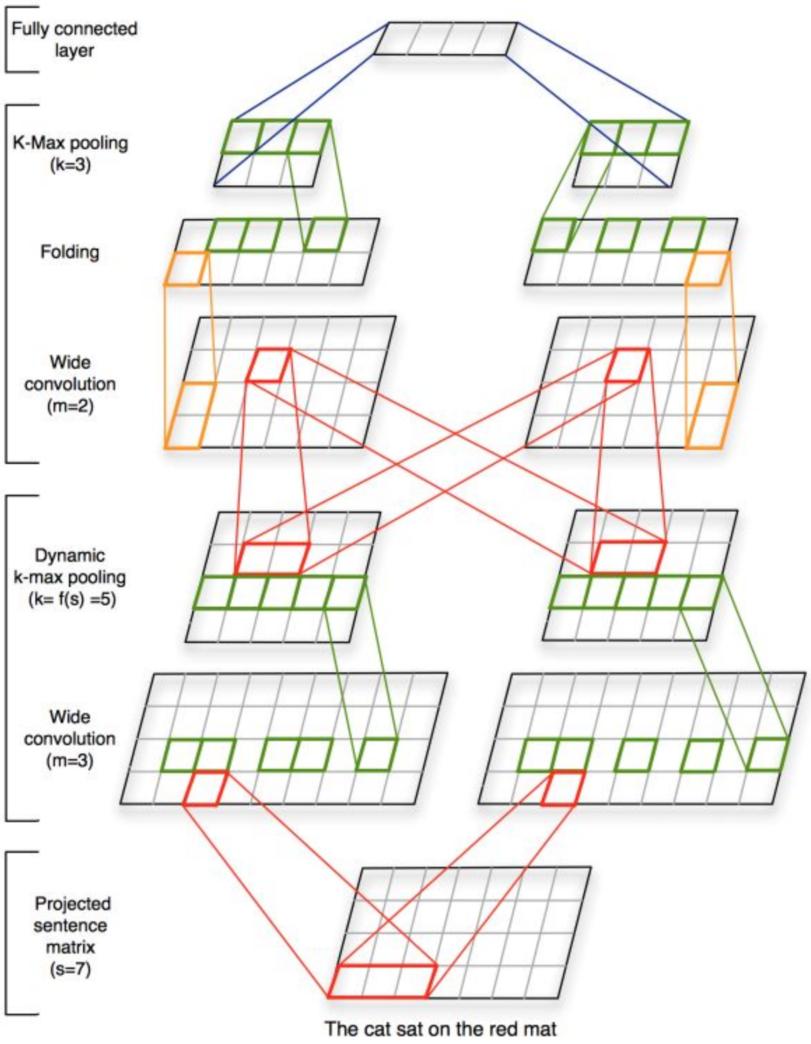
There are three tricky operation here:

- Convolution along rows for 2D filters
- Dynamic k-max pooling
- Folding: sum two rows in one

k for every layer is calculated by the following formula:

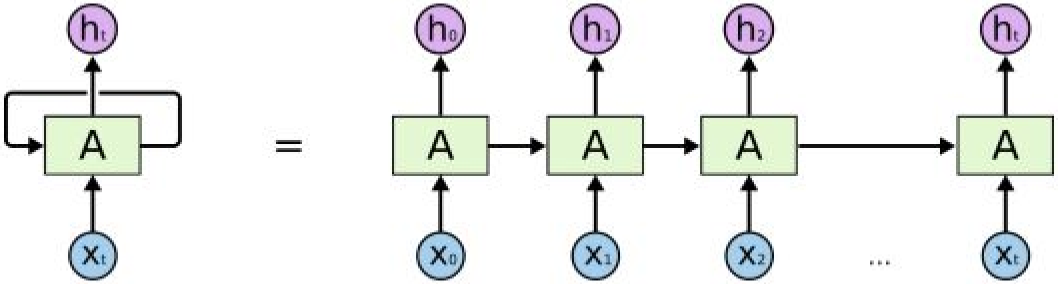
$$k_l = \max(\ k_{top},\ \lceil \frac{L-l}{L} s \rceil\ )$$

http://arxiv.org/pdf/1404.2188v1.pdf



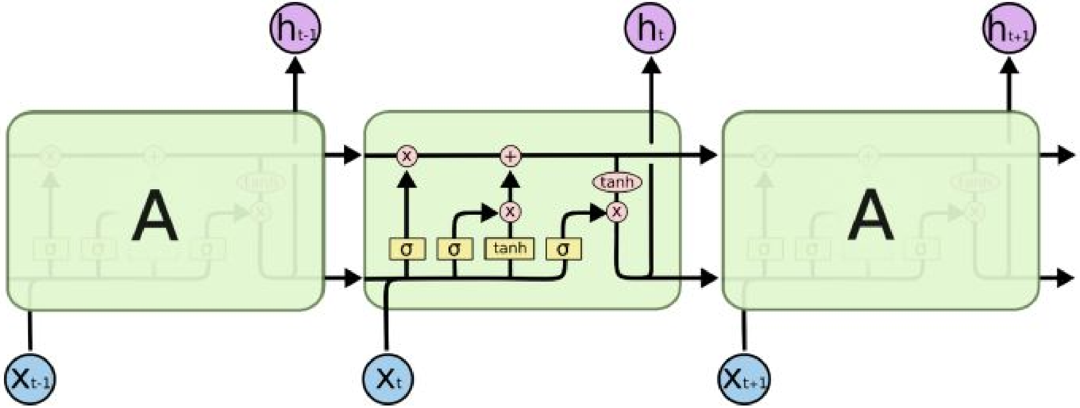
RNN 1: Vanilla RNN

$$h_t = \tanh(W_h h_{t-1} + W_x x_t),$$



An unrolled recurrent neural network.

RNN 2: LSTM



The repeating module in an LSTM contains four interacting layers.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

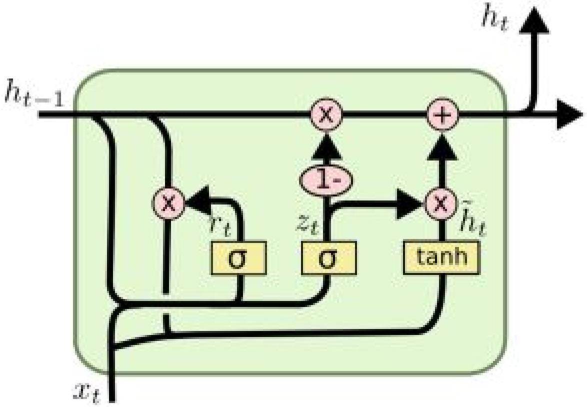
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

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

RNN 3: GRU



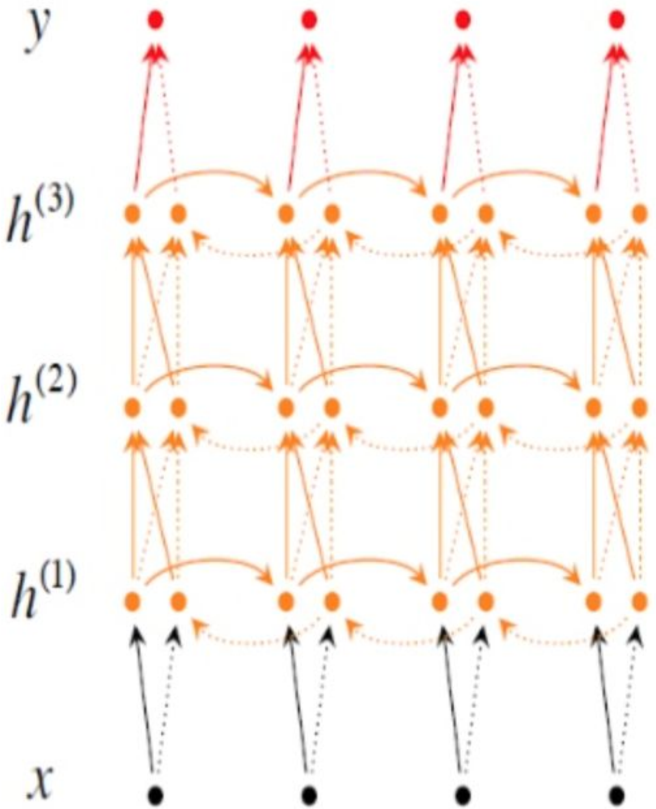
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

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

Multi-layer bidirectional RNN



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

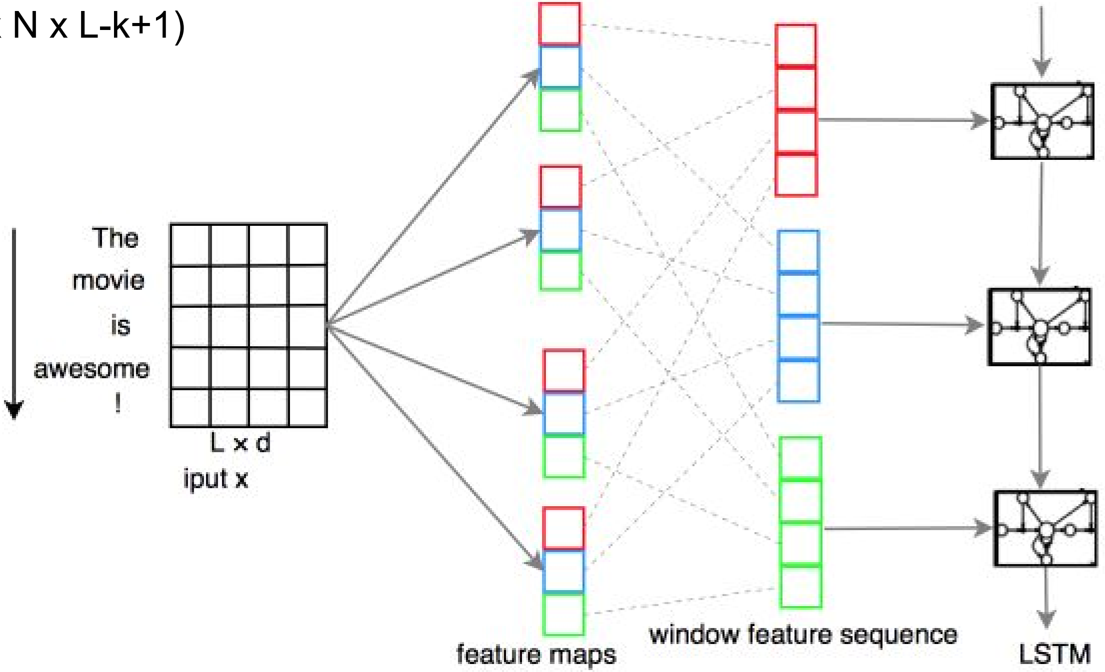
$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

C-LSTM Neural Networks

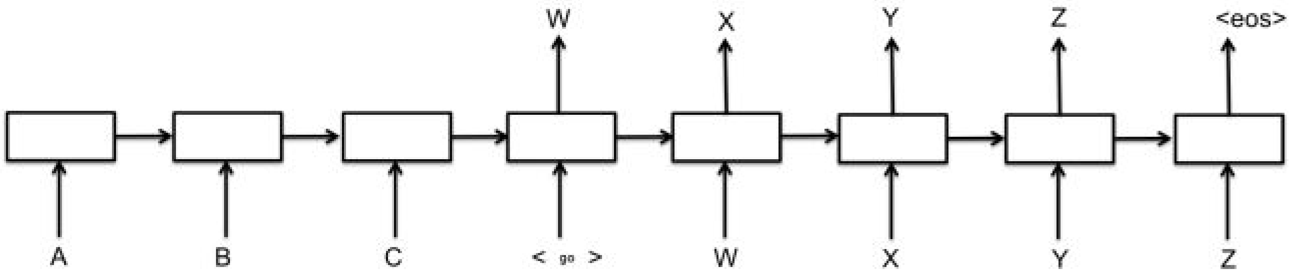
After CNN we have a tensor: (B x N x L-k+1)

, where B - size of mini-batch
N - count of filters
L - max length of sentence
k - length of filters



http://arxiv.org/pdf/1511.08630v2.pdf

Neural machine translation (NMT)

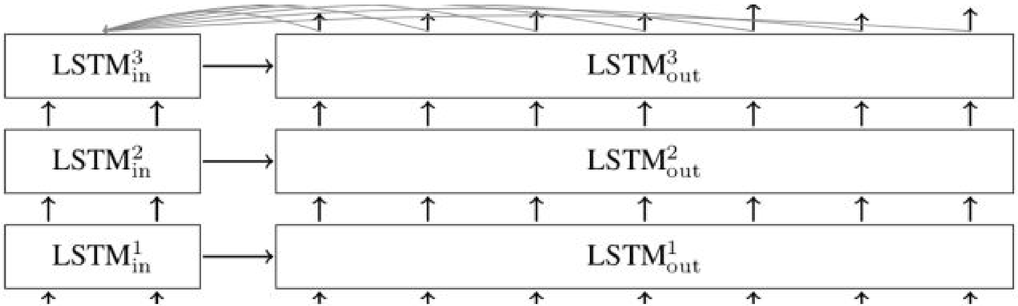


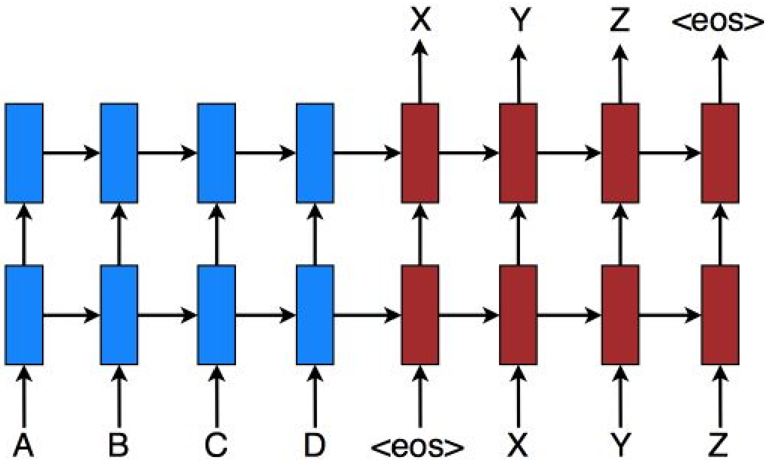
$$1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p(T|S)$$

A,B,C - vectors of words of one language
W,X,Y,Z (below) - vectors of words of another language
W,X,Y,Z (above) - one-hot vectors for another language
<go>, <EOS> - special vectors of the start and end of output sentence
Every rectangular block - LSTM block

http://arxiv.org/pdf/1409.3215v3.pdf

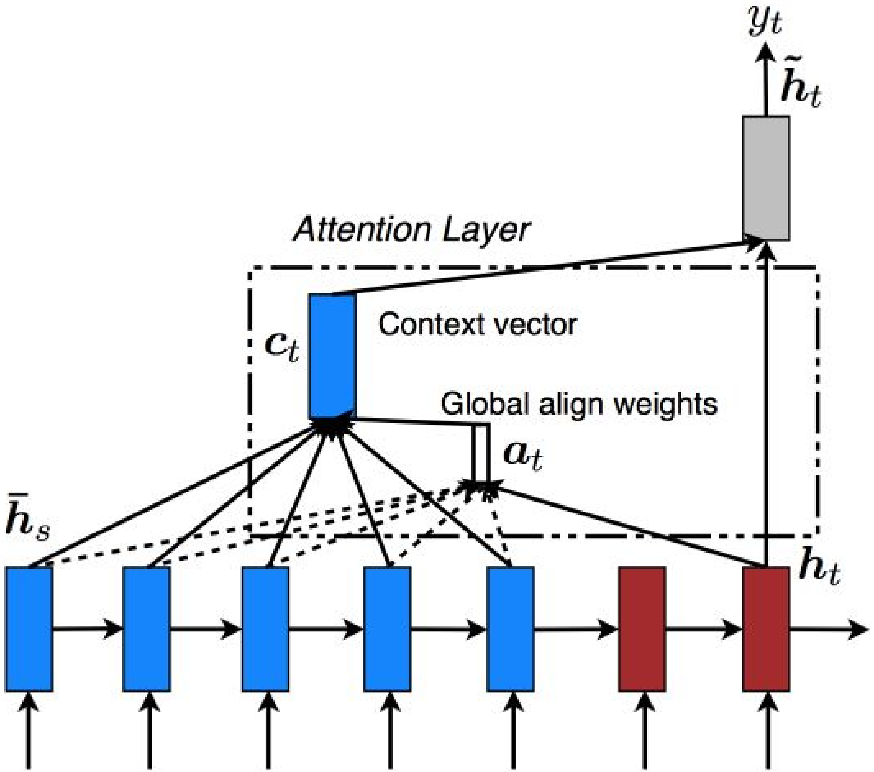
NMT with several layers





The order of the words of the input sentence was reversed!

NMT + ATTENTION



$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s] & \textit{concat} \end{cases}$$

$$\begin{aligned} \boldsymbol{a}_t(s) &= \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \\ &= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \end{aligned}$$

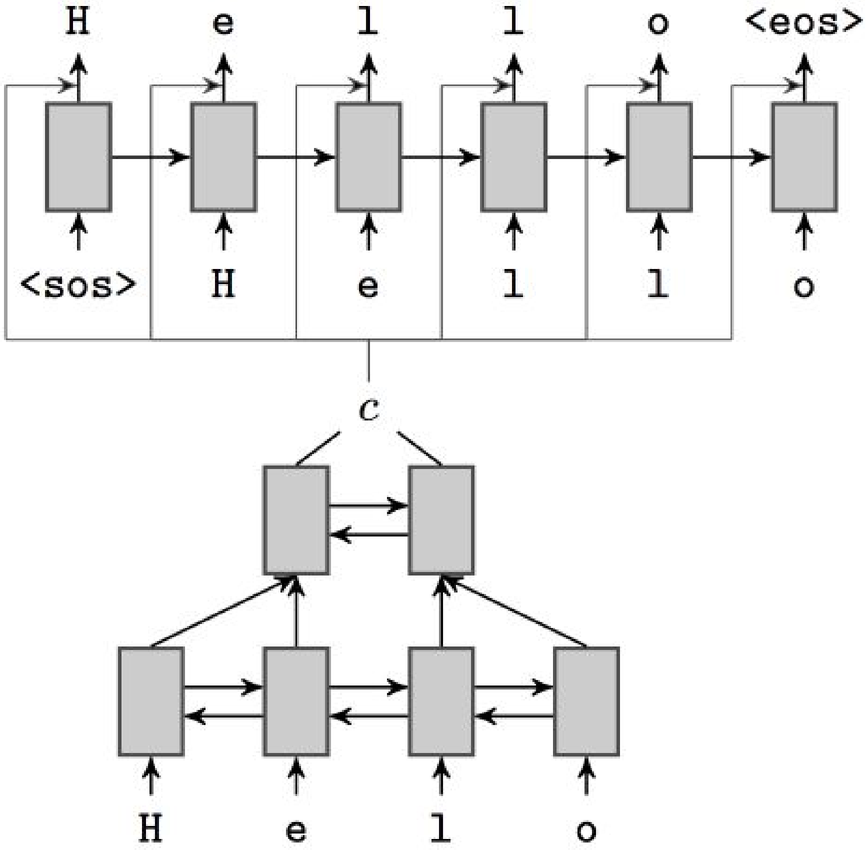
A global context vector c_t is then computed as the weighted average, according to a_t, over all the source states.

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

$$p(y_t | y_{<t}, x) = \text{softmax}(\boldsymbol{W_s} \tilde{\boldsymbol{h}}_t)$$

http://arxiv.org/pdf/1409.0473v6.pdf
http://www.aclweb.org/anthology/D15-1166

Character-level error correction with attention



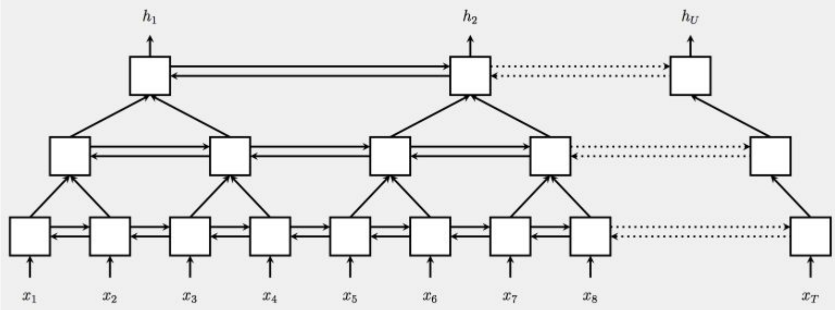
http://arxiv.org/pdf/1603.09727.pdf

Formulas

Encoder

$$f_t^{(j)} = \mathrm{GRU}(f_{t-1}^{(j)}, c_t^{(j-1)}),$$
$$b_t^{(j)} = \mathrm{GRU}(b_{t+1}^{(j)}, c_t^{(j-1)}),$$
$$h_t^{(j)} = f_t^{(j)} + b_t^{(j)}$$

$$c_t^{(j)} = \tanh\left(W_{\text{pyr}}^{(j)}\left[h_{2t}^{(j-1)}, h_{2t+1}^{(j-1)}\right]^\top + b_{\text{pyr}}^{(j)}\right)$$



Decoder

$$d_t^{(j)} = \mathrm{GRU}(d_{t-1}^{(j)}, d_t^{(j-1)}),$$

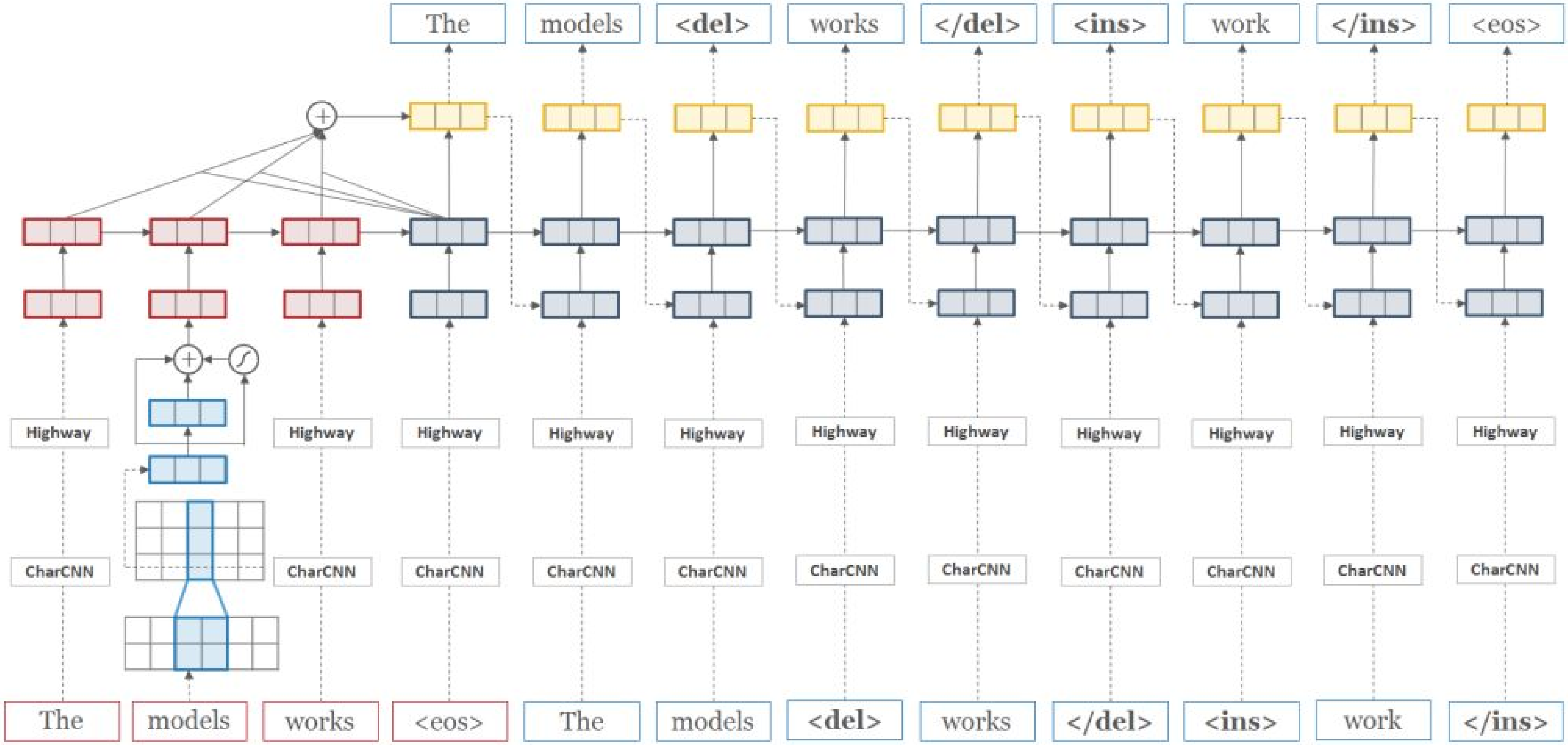
Attention

$$u_{tk} = \phi_1(d_t^{(M)})^\top \phi_2(c_k)$$
$$\alpha_{tk} = \frac{u_{tk}}{\sum_j u_{tj}}$$
$$a_t = \sum_j \alpha_{tj} c_j$$

Output:

The weighted sum of the encoded hidden states $\mathbf{a}_t$ is then concatenated with $\mathbf{d}_{(M)}$, and passed through another affine transform followed by a ReLU nonlinearity before the final softmax output layer.

Sentence-Level Grammatical Error Identification as Sequence-to-Sequence Correction



http://arxiv.org/pdf/1604.04677.pdf

Formulas for word/character level

Encoder with attention to get context vector Cj:

$$u_{j,i} = \mathbf{h}_j^t \cdot \mathbf{W}_\alpha \mathbf{h}_i^s$$

$$\alpha_{j,i} = \frac{\exp u_{j,i}}{\sum_{k \in [1,I]} \exp u_{j,k}}$$

$$\mathbf{v}_j = \sum_{i \in [1,I]} \alpha_{j,i} \mathbf{h}_i^s$$

$$\mathbf{c}_j = \tanh(\mathbf{W}[\mathbf{v}_j ; \mathbf{h}_j^t])$$

Decoder

$$p(t_{j+1} \,|\, \mathbf{s}, \mathbf{t}_{<j}) = \mathrm{softmax}(\mathbf{U}\mathbf{c}_j + \mathbf{b})$$

$$\mathbf{h}_j^t = \mathrm{LSTM}(\mathbf{h}_{j-1}^t, [\mathbf{x}_j^t ; \mathbf{c}_{j-1}])$$

$$\mathbf{h}_0^t \leftarrow \mathbf{h}_I^s$$

CharCNN and Highway Networks

Two separate CharCNNs for Encoder and Decoder:

$$\mathbf{f}_i[k] = \tanh(\langle \mathbf{P}_i[*, k : k + w - 1], \mathbf{H} \rangle + b)$$

$$z_i = \max_k \mathbf{f}_i[k]$$

We use multiple filters H1,...Hh to obtain a vector zi ∈ Rh as the representation for a given source/target word or tag.

Highway network:

$$\hat{\mathbf{z}} = \mathbf{r} \odot f(\mathbf{W}\mathbf{z} + \mathbf{b}) + (\mathbf{1} - \mathbf{r}) \odot \mathbf{z}$$

,where f is ReLu; r = σ(Wrz+br)