Seq2Seq Model

Technische Universität München

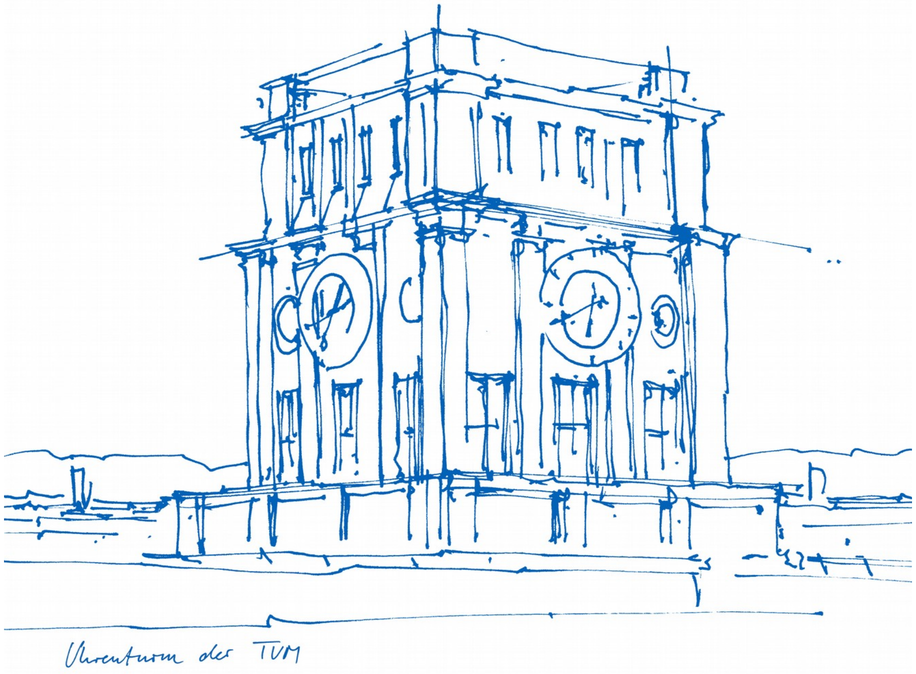
Department of Informatics

Seminar - Applied Deep Learning for NLP

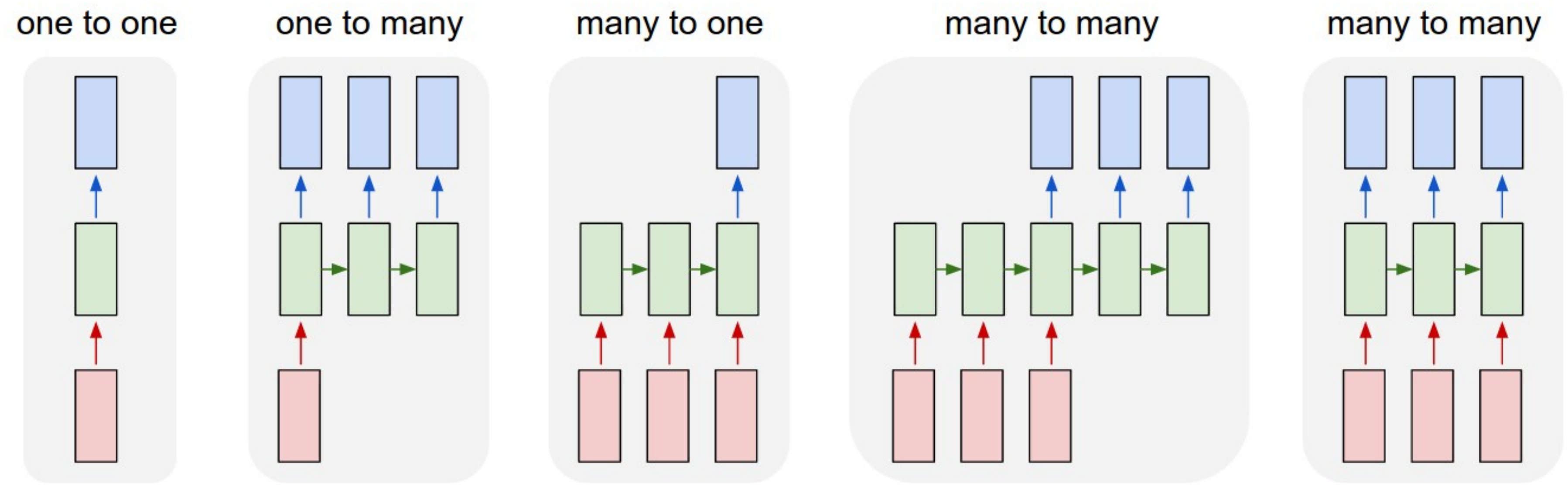
Prof. Dr. Simon Hegelich

Francesco Cognolato

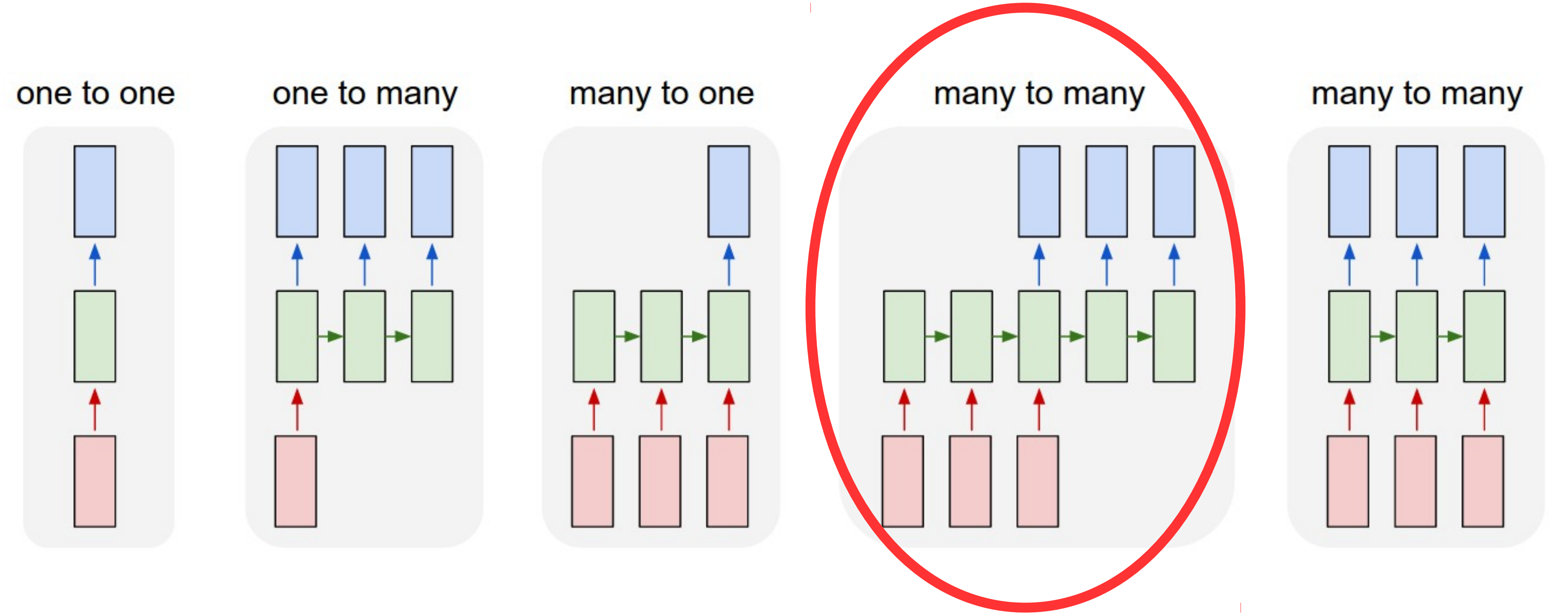
10th January 2019



Different tasks require different models



Different tasks require different models



Seq2Seq

Some possible applications

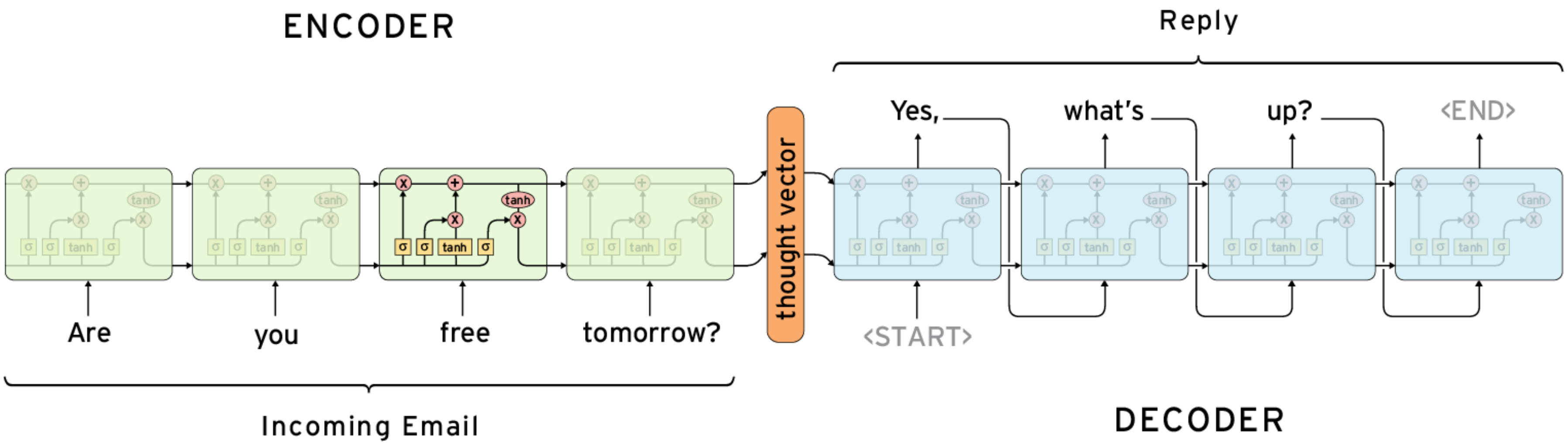
- Neural Machine Translation : input is a sentence (sequence) in one language, output is one in another language

- Text summarization : input is a long text (sequence), output is summary of the text

- Chatbot : input is a sentence, output is a reply

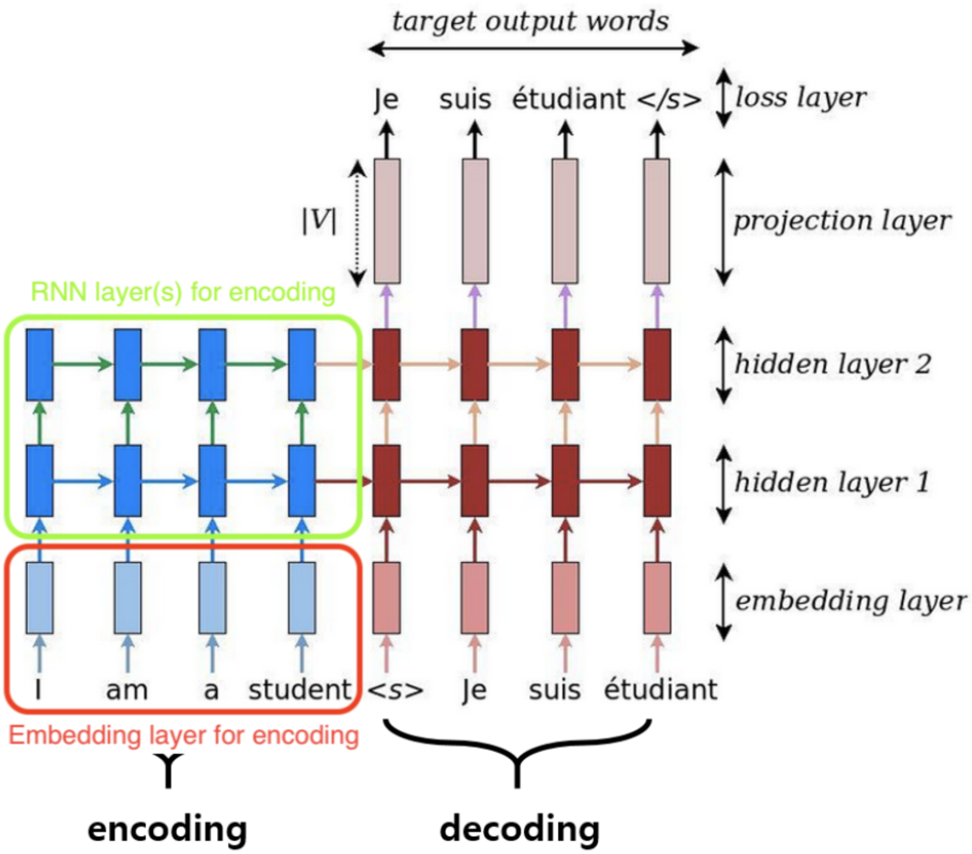
- Speech recognition : input is sequence (digital waveform), output is text

Seq2Seq



ENCODER
Reply
Yes,
what's
up?
<END>
thought vector
Are
you
free
tomorrow?
<START>
Incoming Email
DECODER

Seq2Seq - Basic architecture



Embedding layer

- Nothing more than a matrix, N x F (N words, F embedding size)

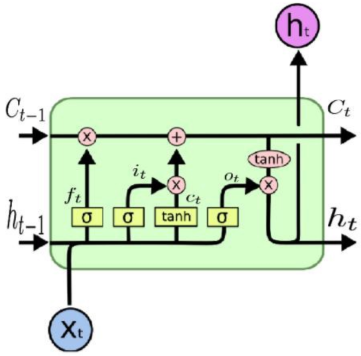
- Both source and target language have their own embedding layer

- Unique mapping between a word and an integer

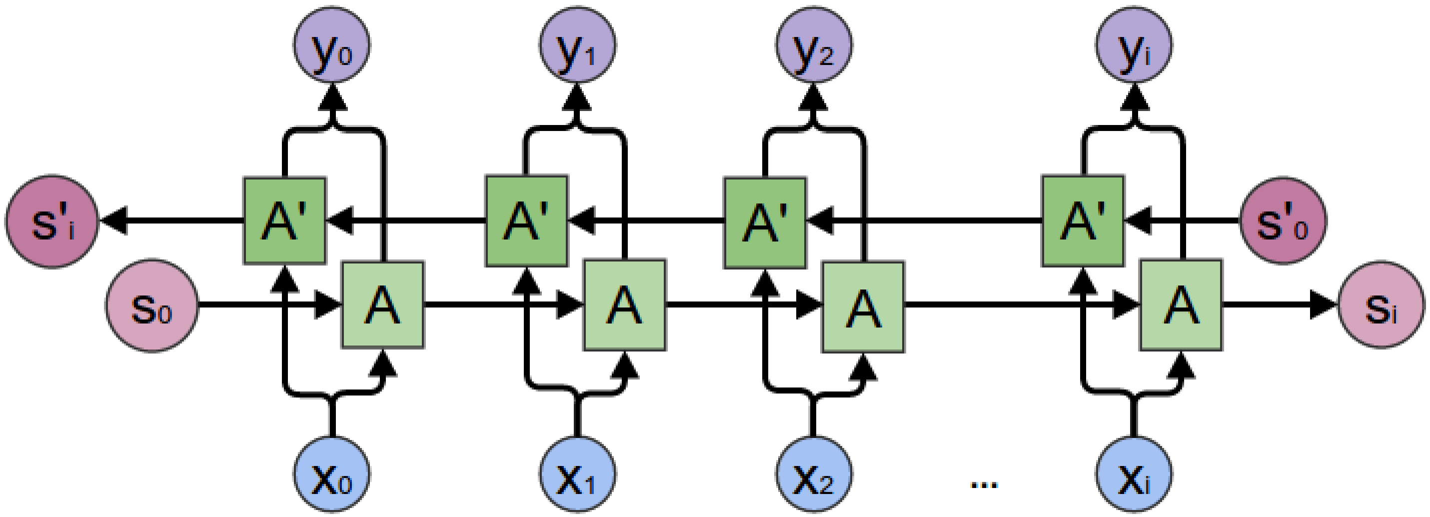
- Look up the word vector given the corresponding row index in the embedding matrix

- If enough data train embeddings from scratch, otherwise use GloVe pretrained embeddings provided from Stanford University

Encoder architecture

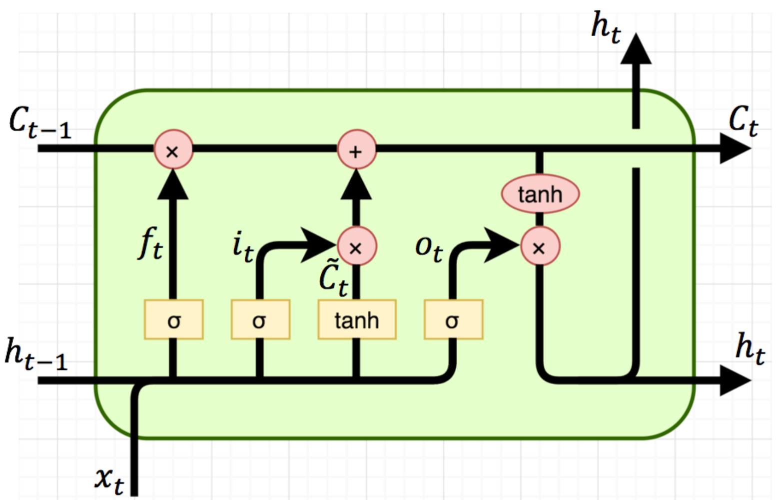


- (Deep) Bidirectional LSTM or GRU

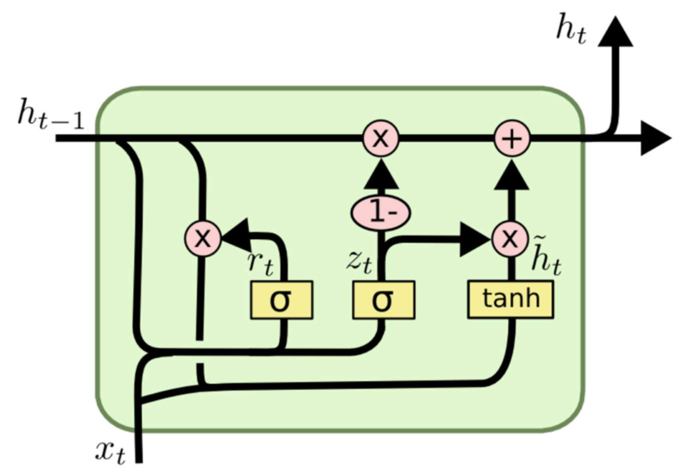


Decoder architecture

- (Deep) LSTM or GRU, initial state is set from last state of encoder



(a) Long Short-Term Memory



(b) Gated Recurrent Unit

Attention mechanism

- In the basic seq2seq architecture, we ignore encoder outputs, but just consider context vector (last state of encoder)

$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \quad \rightarrow \text{ Luong score} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \quad \rightarrow \text{ Bahdanau score*} \end{cases}$$

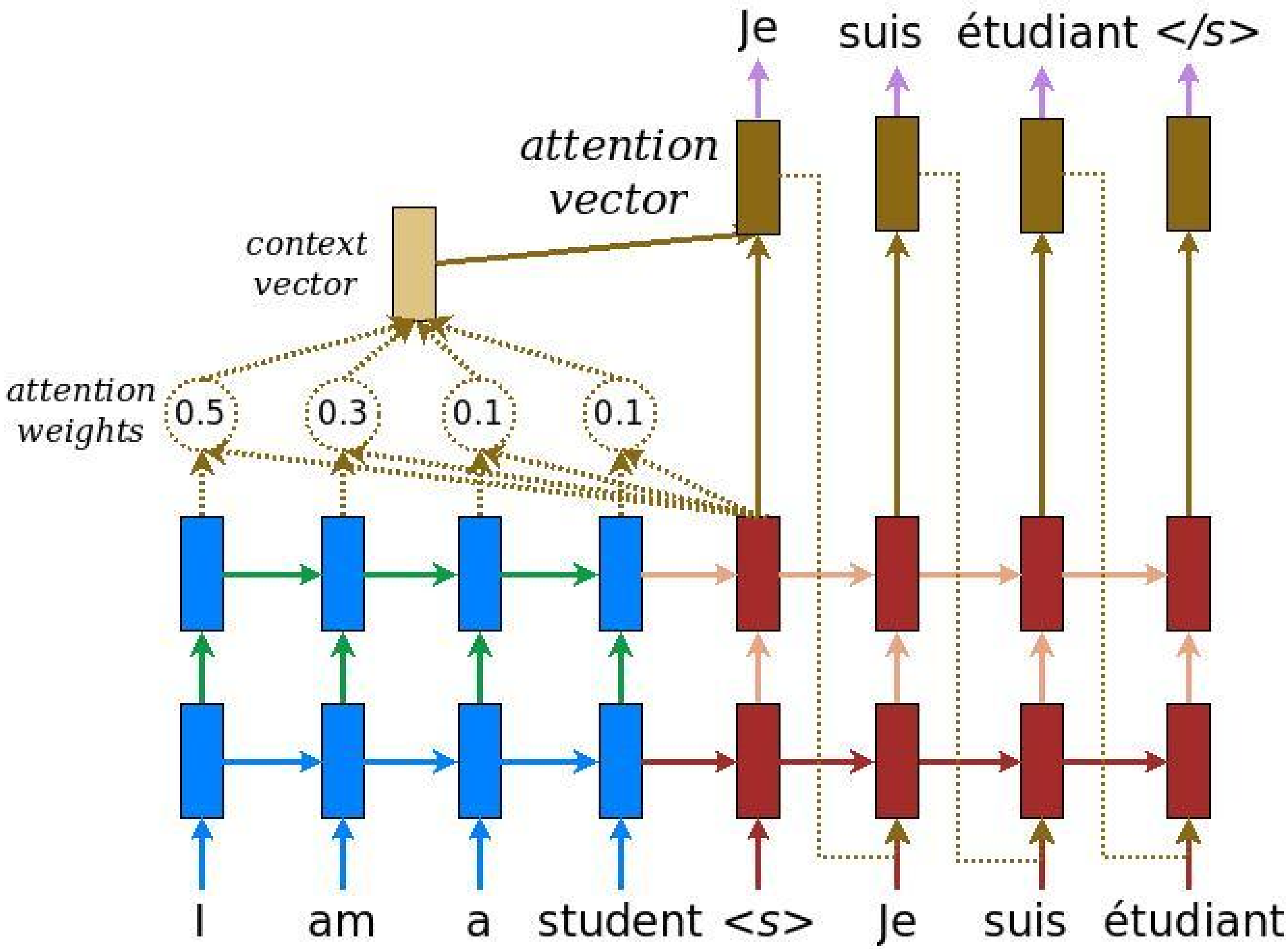
$$\alpha_{ts} = \frac{\exp\left(\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \qquad \text{[Attention weights]}$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \qquad \text{[Context vector]}$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \qquad \text{[Attention vector]}$$

* https://arxiv.org/abs/1409.0473

Attention mechanism



Remarks

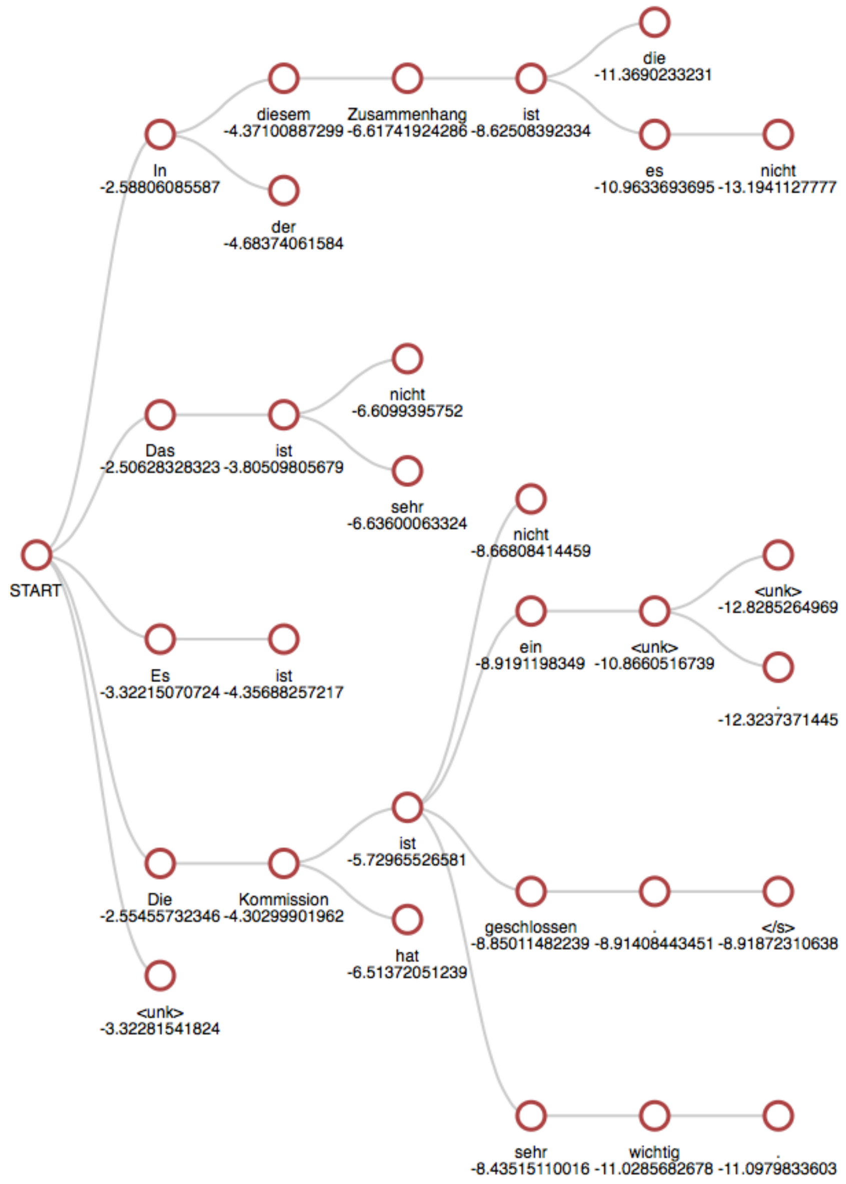
- Add padding of zeros to all the sentences to have the same length (otherwise use some fancy things, like bucketing)

- After decoder, softmax over whole vocabulary and calculate cross entropy loss

- Testing time : source language sequence, decoder input will be only <START> token, generate prediction, append to <START>, generate next token and so on… Stop when <END> token generated (Greedy search)

- Use Beam search, pick best n (e.g. 5-10 beans) predictions and generate a tree of sentences. Final sentence is the one with highest final probabilities

Beam search



START
In
-2.58806085587
diesem
-4.37100887299
Zusammenhang
-6.61741924286
ist
-8.62508392334
die
-11.3690233231
es
-10.9633693695
nicht
-13.1941127777
der
-4.68374061584
Das
-2.50628328323
ist
-3.80509805679
nicht
-6.6099395752
sehr
-6.63600063324
Es
-3.32215070724
ist
-4.35688257217
Die
-2.55455732346
Kommission
-4.30299901962
ist
-5.72965526581
hat
-6.51372051239
nicht
-8.66808414459
ein
-8.9191198349
<unk>
-10.8660516739
<unk>
-12.8285264969
.
-12.3237371445
geschlossen
-8.85011482239
.
-8.91408443451
</s>
-8.91872310638
sehr
-8.43515110016
wichtig
-11.0285682678
.
-11.0979833603
<unk>
-3.32281541824

References

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

- http://karpathy.github.io/2015/05/21/rnn-effectiveness/

- http://web.stanford.edu/class/cs224n/

Questions?

Thank you