Universitat Politècnica de València
Master in Artificial Intelligence, Pattern Recognition and Digital Imaging

2023-2024

MACHINE TRANSLATION

3. Neural Machine Translation

Francisco Casacuberta

fcn@prhlt.upv.es

October 24, 2023

Index

1 Introduction ▷ 2

2 Syncronized feedforward neural networks ▷ 11

3 Syncronized recurrent neural networks ▷ 14

4 Neural machine translation ▷ 23

5 Attention models and recurrent neural networks ▷ 31

6 Attention models for all ▷ 41

7 Training and Decoding with NMT ▷ 67

8 Translation results with neural machine translation ▷ 73

9 Other issues of NMT ▷ 78

10 Bibliography ▷ 90

Index

- ◦ 1 *Introduction* ▷ 2
- 2 Syncronized feedforward neural networks ▷ 11
- 3 Syncronized recurrent neural networks ▷ 14
- 4 Neural machine translation ▷ 23
- 5 Attention models and recurrent neural networks ▷ 31
- 6 Attention models for all ▷ 41
- 7 Training and Decoding with NMT ▷ 67
- 8 Translation results with neural machine translation ▷ 73
- 9 Other issues of NMT ▷ 78
- 10 Bibliography ▷ 90

Introduction

- Traditional approach to MT: Discrete representation of words and sentences.
- Most techniques of machine learning are developped in continuous spaces (i.e. vector spaces)
- In machine learning, (deep) neural networks are good models for many aplications.
- Can words and sentences be represented in a continuous space? Word and sentence embeddings.
- Can neural networks deal with sequences? Dynamic feed-forward networks and recurrent neural networks.

Learned word embeddings

Problem: From words to vectors.

- ***Local (one-hot) coding*** and distributed coding.

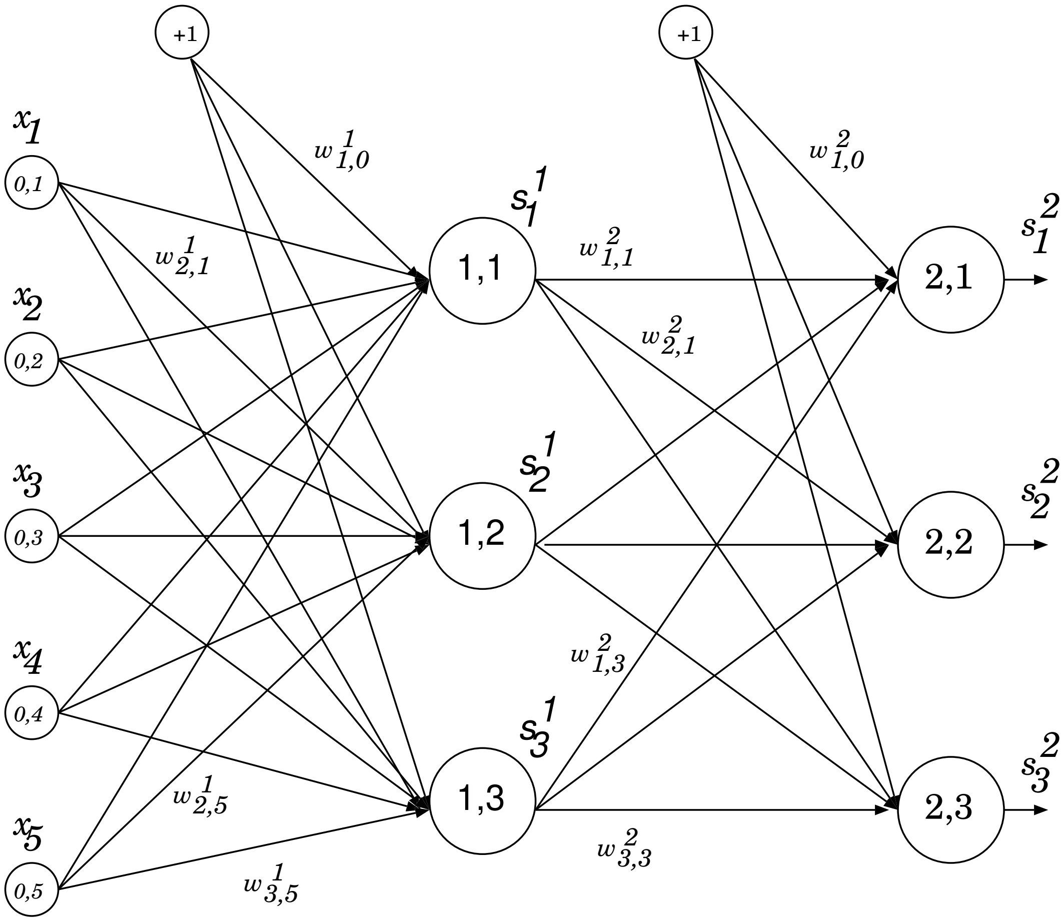
- Neural language model (Bengio 2003)
- Bag-of-words neural networks (Mikolov 2013)
- Continuous skip-grams (Mikolov 2013)
- Pre-trained neural netwotks: BERT (Devlin 2019)
- ...

- Extension to sentence embeddings: SentenceBERT, Universal Sentence Encoder, ...

Byte-Pair Encoding (BPE) (Sennrich 2015)

- Problem: Word embedding of unseen words.
- Solution: The use of subword units learned from a training set.
- *Byte Pair Encoding* (BPE) is a data compression technique that iteratively replaces the most frequent pair of bytes in a sequence with a single, unused byte.
- Toolkit: `https://github.com/rsennrich/subword-nmt`
- Other related techniques: SentencePiece, ...
- Character based units: convolutational networks.

A multilayer perceptron



Hidden layer Output layer

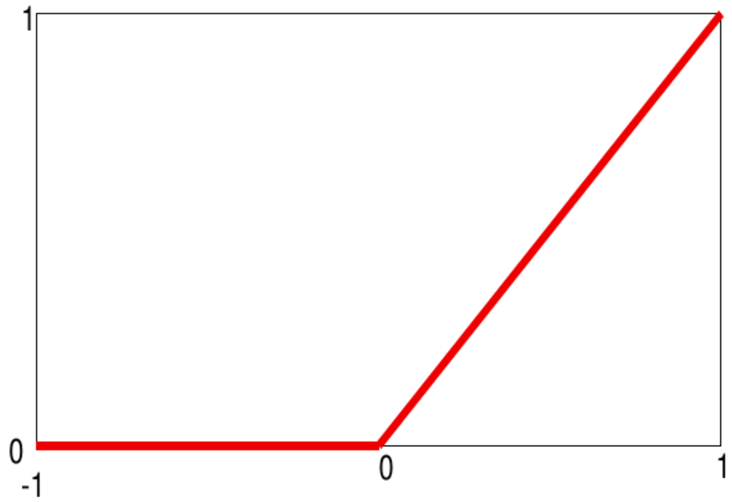
$\mathbf{s}^1 = \mathbf{f}(\mathbf{W}^1\, \mathbf{x})$ $\mathbf{s}^2 = \mathbf{f}(\mathbf{W}^2\, \mathbf{s}^1)$

Activation functions

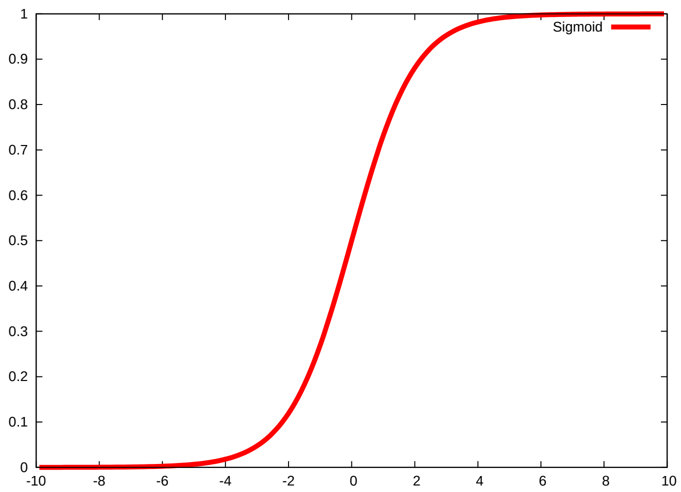
$$\mathbf{f}(\mathbf{z}) = (f(z_1), \ldots, f(z_d))^t \text{ for } \mathbf{z} \equiv (z_1, \ldots, z_d)^t \in \mathbb{R}^d$$

- ***ReLU*** (rectified linear unit): $f_R(z_j) = \max(0, z_j)$
- ***Sigmoid***: $f_S(z_j) = \frac{1}{1+\exp(-z_j)}$
- ***Hyperbolic tangent***: $f_T(z_j) = \frac{\exp(z_j)-\exp(-z_j)}{\exp(z_j)+\exp(-z_j)}$ $\quad (f_T(z_j) = 2\, f_S(2\, z_j) - 1)$
- ***Softmax:*** $f_{SM}(z_j) = \frac{\exp(z_j)}{\sum_{j'} \exp(z_{j'})}$ $\quad \left(f_{SM}(z_j) = f_S(z_j - \ln(\sum_{j' \neq j} \exp(z_{j'})) \right)$
- PReLU, ELU, Maxout, ...

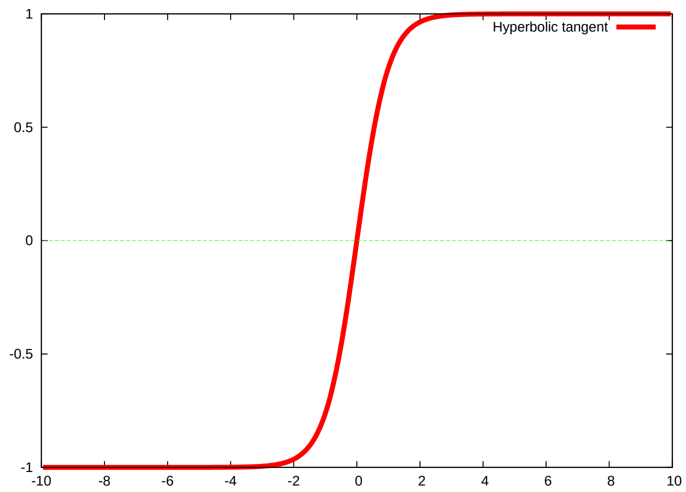
Activation functions



ReLu



Sigmoid



Hyperbolic tangent

The softmax activation function

A problem with the use of softmax in training

$$f_{sm}(z_w) = \frac{\exp(z_w)}{\sum_{w'} \exp(z_{w'})}$$

The sum in the normalization factor is extended on all the vocabulary $|V|$.

Solutions:

- Hierarchical softmax: using a binary tree representation of the output layer (Huffman coding)
- Negative sampling: each training sample only modify a subset of the weights (the weights of a small subset of “negative” words).
- Subsampling of frequent words.

Training neural networks

- A neural network defines $\mathbf{f} : \mathbb{R}^{D_X} \rightarrow \mathbb{R}^{D_Y}$: $\mathbf{f}(\mathbf{x}_n; \mathbf{W}) = \mathbf{y}_n$ for $1 \leq n \leq N$.
- A ***training sequence*** $T = (\mathbf{x}_1, \mathbf{t}_1), \ldots, (\mathbf{x}_N, \mathbf{t}_N) : \mathbf{x}_n \in \mathbb{R}^{D_X}$, $\mathbf{t}_n \in \mathbb{R}^{D_Y}$
- ***Total error*** (regression) is: $\mathcal{F}_T(\mathbf{W}) = \sum_{n=1}^{N} \frac{1}{2} \sum_{i=1}^{D_Y} (t_{n,i} - y_{n,i})^2$
- ***Cross-entropy loss*** (classification) ($\mathbf{t}_n \in \{0,1\}^{D_Y}$): $\mathcal{F}_T(\mathbf{W}) = -\sum_{n=1}^{N} \sum_{i=1}^{D_Y} t_{n,i} \log y_{n,i}$
- Training goal: $\widehat{\mathbf{W}} = \operatorname*{argmin}_{\mathbf{W}} \mathcal{F}_T(\mathbf{W})$
- Computing a local minimum of $\mathcal{F}$: GRADIENT DESCENT

$$\Delta \mathbf{W} = -\rho \nabla_{\mathbf{W}} \mathcal{F}_T(\mathbf{W})$$

Index

1 Introduction ▷ 2

○ 2 *Syncronized feedforward neural networks* ▷ 11

3 Syncronized recurrent neural networks ▷ 14

4 Neural machine translation ▷ 23

5 Attention models and recurrent neural networks ▷ 31

6 Attention models for all ▷ 41

7 Training and Decoding with NMT ▷ 67

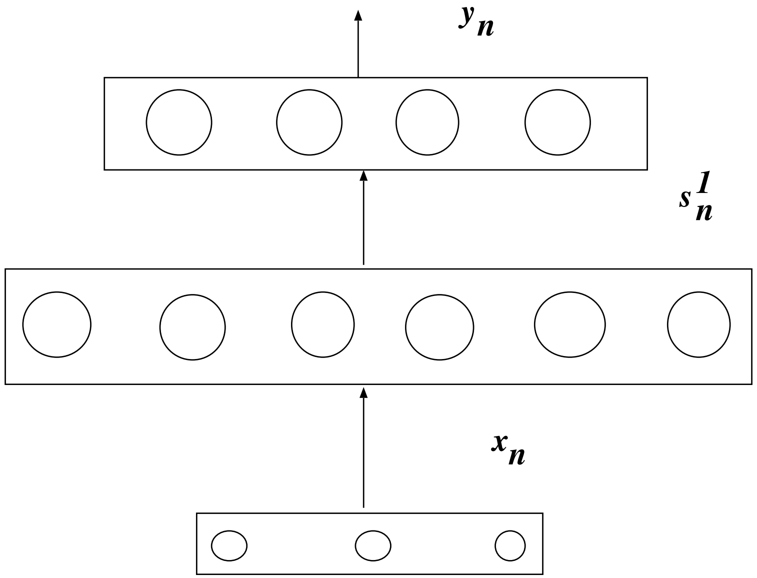
8 Translation results with neural machine translation ▷ 73

9 Other issues of NMT ▷ 78

10 Bibliography ▷ 90

Sequence processing with dynamic feedforward networks

$$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \rightarrow \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$$

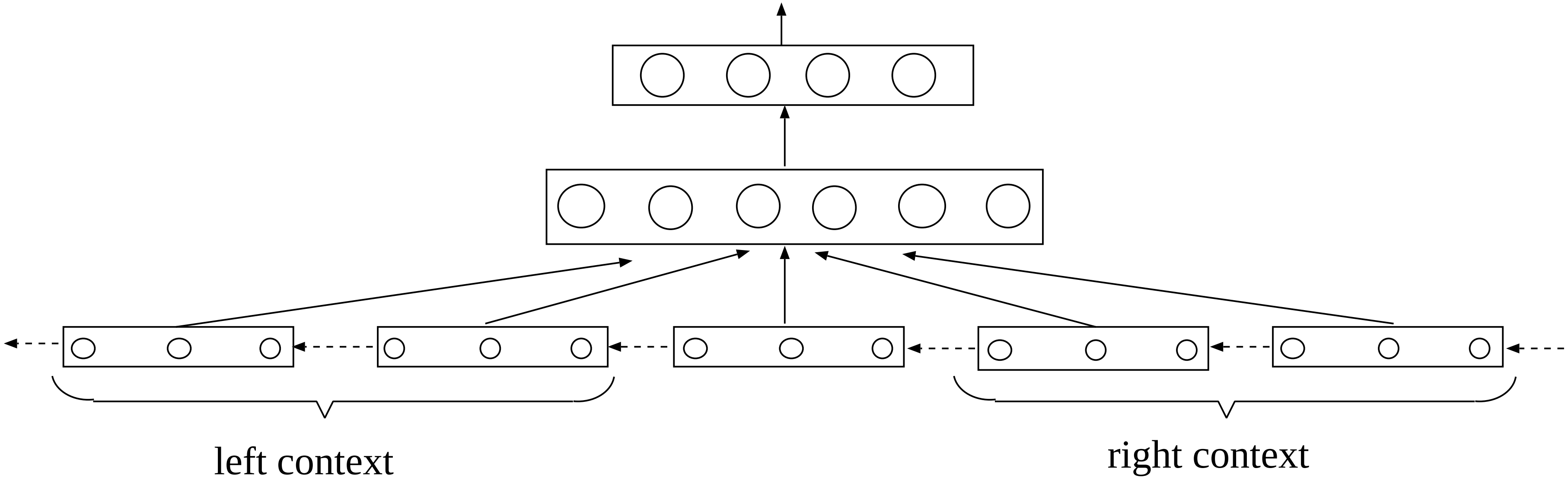


$$1 \leq n \leq N$$

t	input layer		hidden layer		output layer
1	$\mathbf{x}_1$	$\Rightarrow$	$\mathbf{s}_1^1$	$\Rightarrow$	$\mathbf{y}_1$
2	$\mathbf{x}_2$	$\Rightarrow$	$\mathbf{s}_2^1$	$\Rightarrow$	$\mathbf{y}_2$
			$\ldots$		
N	$\mathbf{x}_N$	$\Rightarrow$	$\mathbf{s}_N^1$	$\Rightarrow$	$\mathbf{y}_N$

Sequence processing with dynamic feedforward networks

$$\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N \rightarrow \mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N$$



$$1 \leq n \leq N$$

t	input layer		hidden layer		output layer
1	$\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$	$\Rightarrow$	$\mathbf{s}_1^1$	$\Rightarrow$	$\mathbf{y}_1$
2	$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$	$\Rightarrow$	$\mathbf{s}_2^1$	$\Rightarrow$	$\mathbf{y}_2$
			...		
N	$\mathbf{x}_{N-1}, \mathbf{x}_N, \mathbf{x}_{N+1}$	$\Rightarrow$	$\mathbf{s}_N^1$	$\Rightarrow$	$\mathbf{y}_N$

Index

1 Introduction ▷ 2

2 Syncronized feedforward neural networks ▷ 11

∘ 3 *Syncronized recurrent neural networks* ▷ 14

4 Neural machine translation ▷ 23

5 Attention models and recurrent neural networks ▷ 31

6 Attention models for all ▷ 41

7 Training and Decoding with NMT ▷ 67

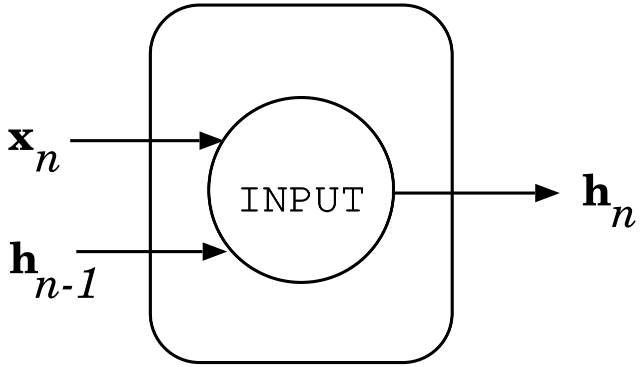
8 Translation results with neural machine translation ▷ 73

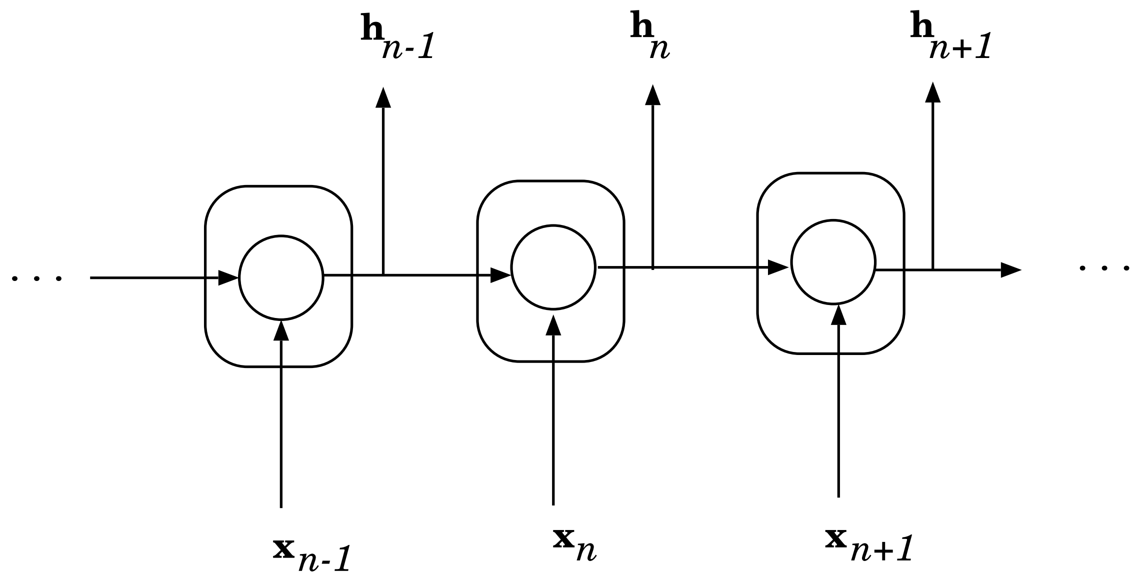
9 Other issues of NMT ▷ 78

10 Bibliography ▷ 90

Simple recurrent networks

- A set H of D_H units and a set X of D_X inputs.
- Input: $\mathbf{x}_n \in \mathbb{R}^{D_X}$ and output: $\mathbf{h}_n \in \mathbb{R}^{D_H}$, $n = 1, 2, \ldots$
- $\mathbf{y}_n = \mathbf{h}_n = \mathbf{f}(\mathbf{W}_X\, \mathbf{x}_n + \mathbf{W}_H\, \mathbf{h}_{n-1}) = \mathbf{F}(\mathbf{x}_n, \mathbf{h}_{n-1})$ with $\mathbf{h}_0 = \mathbf{0}$

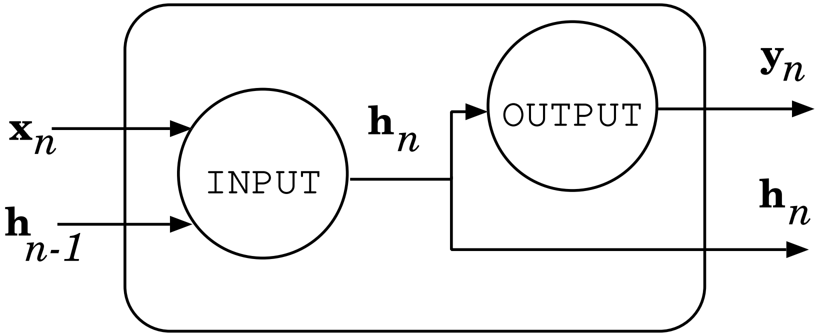




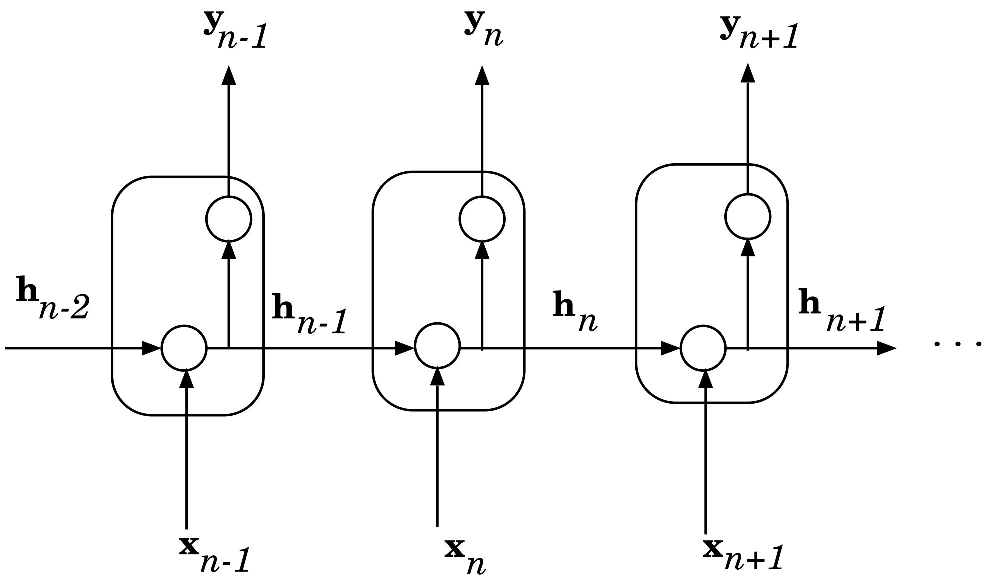
Unfolding a simple recurrent network

Elman recurrent neural networks

- A set Y of D_Y units, a set X of D_X inputs and a set H of D_H hidden units
- Input: $\mathbf{x}_n \in \mathbb{R}^{D_X}$, output: $\mathbf{y}_n \in \mathbb{R}^{D_Y}$, and hidden states: $\mathbf{h}_n \in \mathbb{R}^{D_H}$, $n = 1, 2, \ldots$
- $\mathbf{y}_n = \mathbf{f}(\mathbf{W}_Y\ \mathbf{h}_n)$ and $\mathbf{h}_n = \mathbf{f}(\mathbf{W}_H\ \mathbf{h}_{n-1} + \mathbf{W}_X\ \mathbf{x}_n)$ with $\mathbf{h}_0 = \mathbf{0}$



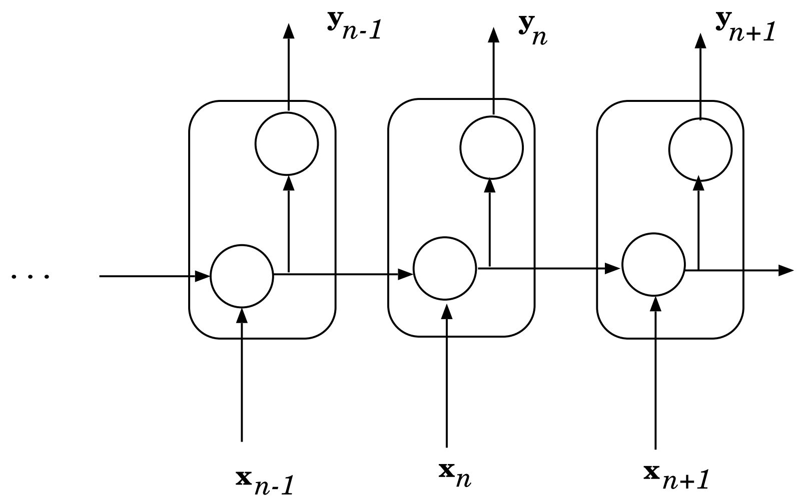
. . .



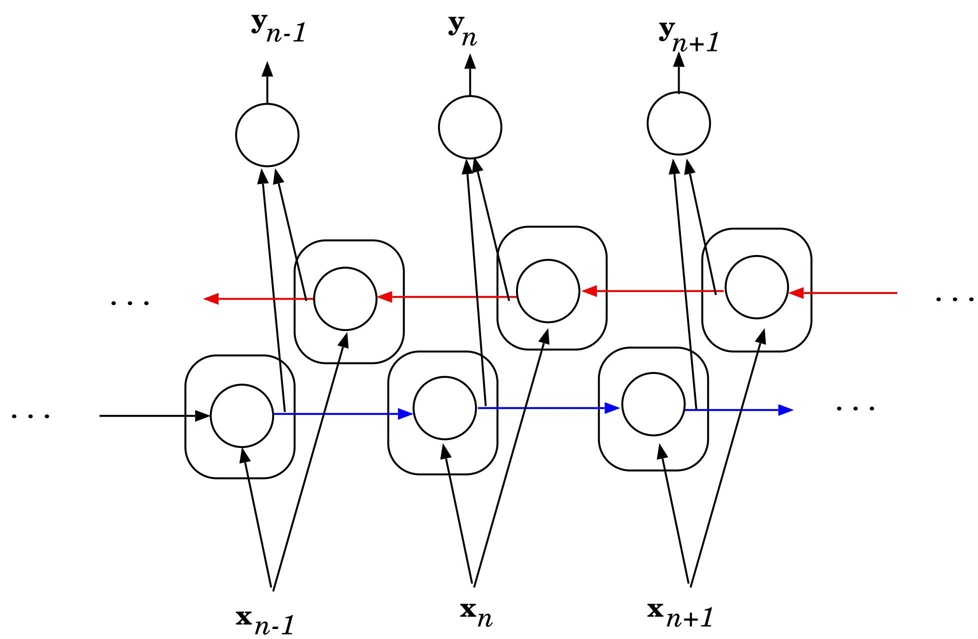
Unfolding Elman network

Bidirectional recurrent neural networks (Schuster & Paliwal 1997)

Dependencies with the past inputs, but with the future?



Standard dynamics



Bidirectional dynamics

Bidirectional recurrent neural networks

- A set of D_H forward units, a set of D_H backward units, a set of D_Y output units and a set of D_X inputs.

- Input: $\mathbf{x}_n \in \mathbb{R}^{D_X}$ and output: $\mathbf{y}_n \in \mathbb{R}^{D_Y}$.,

- ***Forward***: $\mathbf{h}^f_n = \mathbf{f}(\mathbf{W}^f_H\, \mathbf{h}^f_{n-1} + \mathbf{W}^f_X\, \mathbf{x}_n) \quad n = 1, 2, \ldots, N$ with $\mathbf{h}_0 = \mathbf{0}$

- ***Backward***: $\mathbf{h}^b_n = \mathbf{f}(\mathbf{W}^b_H\, \mathbf{h}^b_{n+1} + \mathbf{W}^b_X\, \mathbf{x}_n) \quad n = N, N-1, \ldots, 1$ with $\mathbf{h}_{N+1} = \mathbf{0}$

- ***Output***: $\mathbf{y}_n = \mathbf{f}(\mathbf{W}^f_Y \mathbf{h}^f_n + \mathbf{W}^b_Y \mathbf{h}^b_n) \qquad n = 1, 2, \ldots, N$

- ***Output*** (alternative): $\mathbf{y}_n = \mathbf{f}(\mathbf{W}_Y [\mathbf{h}^f_n, \mathbf{h}^b_n]) \qquad n = 1, 2, \ldots, N$

Training recurrent neural networks

- Back-propagation through time algorithm (BPTT)
- Back-propagation through time algorithm with momentum.
- Truncate gradiente with momentum.

Text translation using Elman networks

(Castaño et al. Eurospeech 1997)

- ***Problem***: translate a source sentence from a source language to a sentence from a target language in a hotel-desk scenario:

  /¿Tiene habitaciones libres?/ ⇒ /Do you have any room available?/

- ***Topology***
  - Elman network
  - Distributed coding:
    - 61 input units (132 words)
    - 52 output units (87 words)
    - 160 hidden units
  - windows of 6 words

- ***Training***
  - Truncate gradiente with momentum and pattern-based training
  - Training corpus: 5000 labeled pairs
  - Number of iterations: 100.

- ***Test***
  - 1000 pairs (12.6 words / source sentence and 11.8 words / output sentence).
  - Percentage of right translated sentences: 49.5%
  - Percentage of right translated words: 93.1%

Long Short-Term Memory (LSTM) [Hochreiter 1997]

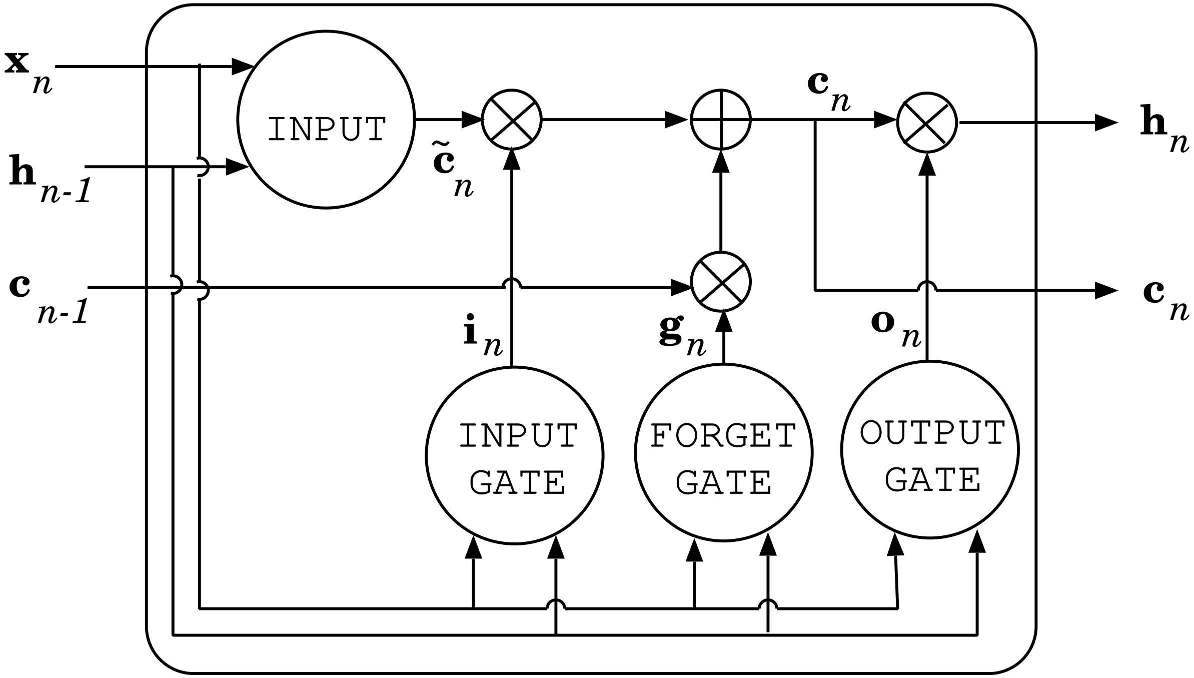
Problems with recurrent neural networks:

- The output depends on the complete past information and sometimes only recent information is needed and sometimes more far context is needed

- In back-propagation through time algorithm and exact gradient or real-time recurrent learning algorithms the errors that propagated backwards in time tend to vanish or to oscillate.

A solution: Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU)

Long Short-Term Memory (LSTM)



- $\mathbf{i}_n = \mathbf{f}_s(\mathbf{W}_Y^I \mathbf{h}_{n-1} + \mathbf{W}_X^I \mathbf{x}_n)$
- $\mathbf{g}_n = \mathbf{f}_s(\mathbf{W}_Y^F \mathbf{h}_{n-1} + \mathbf{W}_X^F \mathbf{x}_n)$
- $\mathbf{o}_n = \mathbf{f}_s(\mathbf{W}_Y^O \mathbf{h}_{n-1} + \mathbf{W}_X^O \mathbf{x}_n)$
- $\tilde{\mathbf{c}}_n = \mathbf{f}_{th}(\mathbf{W}_Y^C \mathbf{h}_{n-1} + \mathbf{W}_X^C \mathbf{x}_n)$
- $\mathbf{c}_n = \mathbf{g}_n \times \mathbf{c}_{n-1} + \mathbf{i}_n \times \tilde{\mathbf{c}}_n$
- $\mathbf{h}_n = \mathbf{o}_n \times \mathbf{f}_{th}(\mathbf{c}_n)$

$$\mathbf{h}_n \;=\; \mathbf{F}(\mathbf{x}_n, \mathbf{h}_{n-1})$$

Index

1 Introduction ▷ 2

2 Syncronized feedforward neural networks ▷ 11

3 Syncronized recurrent neural networks ▷ 14

◦ 4 *Neural machine translation* ▷ 23

5 Attention models and recurrent neural networks ▷ 31

6 Attention models for all ▷ 41

7 Training and Decoding with NMT ▷ 67

8 Translation results with neural machine translation ▷ 73

9 Other issues of NMT ▷ 78

10 Bibliography ▷ 90

Neural Machine Translation

- Early proposals with syncronized recurrent neural networks (Castaño 1997)

- For a source sentence x_1^J search for a target sentence $\hat{y}_1^{\hat{I}}$,

$$
\begin{aligned}
\hat{y}_1^{\hat{I}} &= \underset{I, y_1^I}{\text{argmax}}\, p(y_1^I \mid x_1^J) \\
&= \underset{I, y_1^I}{\text{argmax}} \prod_{i=1}^{I} p(y_i \mid y_1^{i-1}, x_1^J) \\
&= \underset{I, y_1^I}{\text{argmax}} \prod_{i=1}^{I} p(y_i \mid y_1^{i-1}, u(x_1^J))
\end{aligned}
$$

where $u(x_1^J)$ is a representation of x_1^J.

Neural Machine Translation

- Non syncronized neural networks (many available tool-kits): The encoder-decoder approach (+ attention models) based on
  - LSTMs or GRUs + cross-attention model.
  - Transformer (attention models for everything).
  - Convolutional neural networks and/or recurrent neural networks (hybrid).
- MT applications:
  - Text and speech translation.
  - Interactive MT, multimodal MT, automatic post-editing, modernization of old text transcriptions, ...
- Other applications:
  - Automatic summarization.
  - Image description, video description, multilingual image description, visual question answering, image generation,
  - ...

Encoder-decoder approach to NMT (Peris 2018)

1. Project source words to a sequence of continuous vectors $\rightarrow$ Source word embeddings.
2. Generate a contextual representation of the source sentence $\rightarrow$ Encoder NN.
3. Generate an a-posteriori probabilistic distributions of succesive target words using the contextual representation of the source words $\rightarrow$ Decoder NN.
4. Encode target words $\rightarrow$ Target word embeddings.

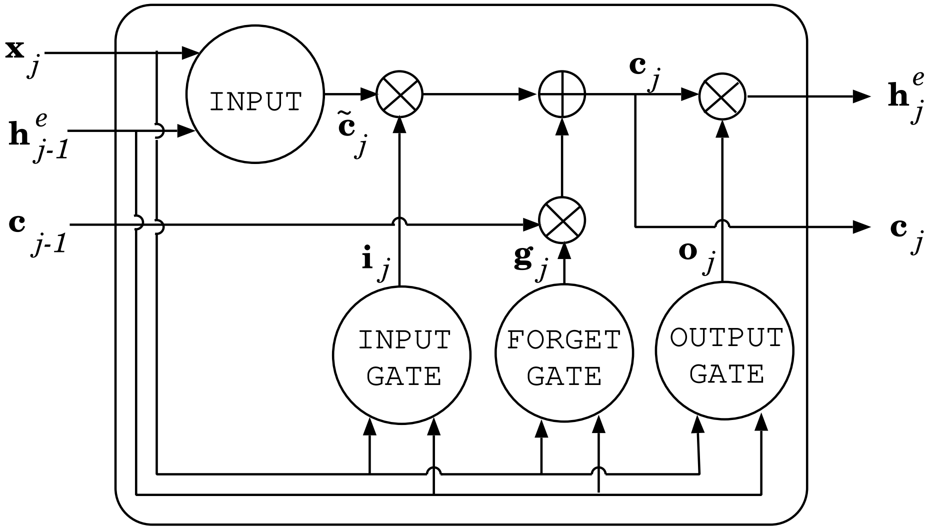
Encoder-decoder with LSTMs only

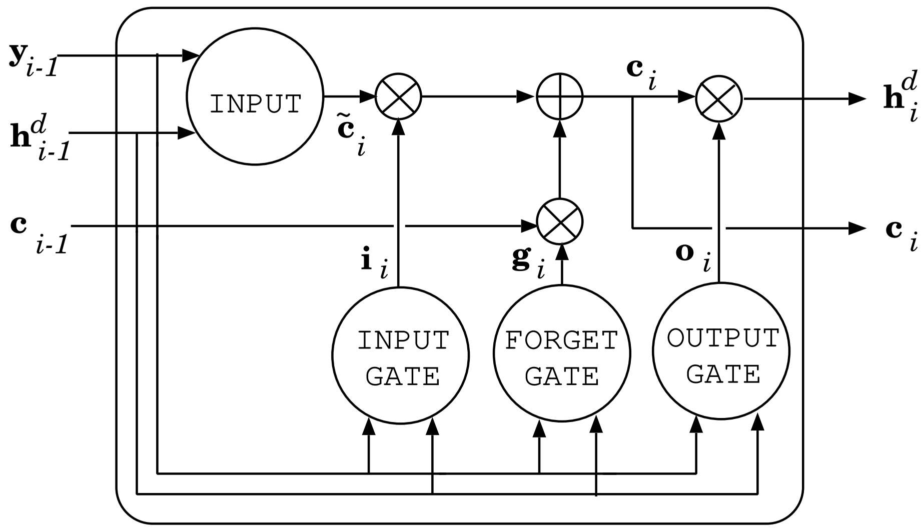
- Goal: Given a source sentence x_1^J and a target sentence y_1^I, compute:

$$p(y_1^I \mid x_1^J) = \prod_{i=1}^{I} p(y_i \mid y_1^{i-1}, u(x_1^J))$$

- Neural networks: LSTMs or GRUs

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





$$\mathbf{h}^e_j \;=\; \mathbf{F}(\mathbf{x}_j, \mathbf{h}^e_{j-1}) \quad 1 \le j \le J$$

$$\mathbf{h}^d_i \;=\; \mathbf{F}(\mathbf{y}_{i-1}, \mathbf{h}^d_{i-1}) \quad 1 \le i \le I$$

Encoder-decoder approach to NMT (Bahdanau et al. 2015)

- The RNN-based encoder ($\mathbf{F}_e$) converts a source word sequence $x_1, \ldots, x_J \equiv x_1^J$ into a vector $\mathbf{u} = \mathbf{u}(x_1^J)$,

$$\mathbf{h}_0^e = \mathbf{0}$$

$$\mathbf{h}_j^e = \mathbf{F}_e(\mathbf{W}_E(x_j), \mathbf{h}_{j-1}^e) = \mathbf{F}_e(\mathbf{x}_j, \mathbf{h}_{j-1}^e) \quad 1 \leq j \leq J$$

$$u(x_1^J) \equiv \mathbf{u}(\mathbf{h}_1^e, \ldots, \mathbf{h}_J^e) = \mathbf{u}$$

  where $\mathbf{h}_j^e$ is the states of a RNN-encoder and $\mathbf{u}$ could be $\mathbf{h}_J^e$ (sentence embedding)

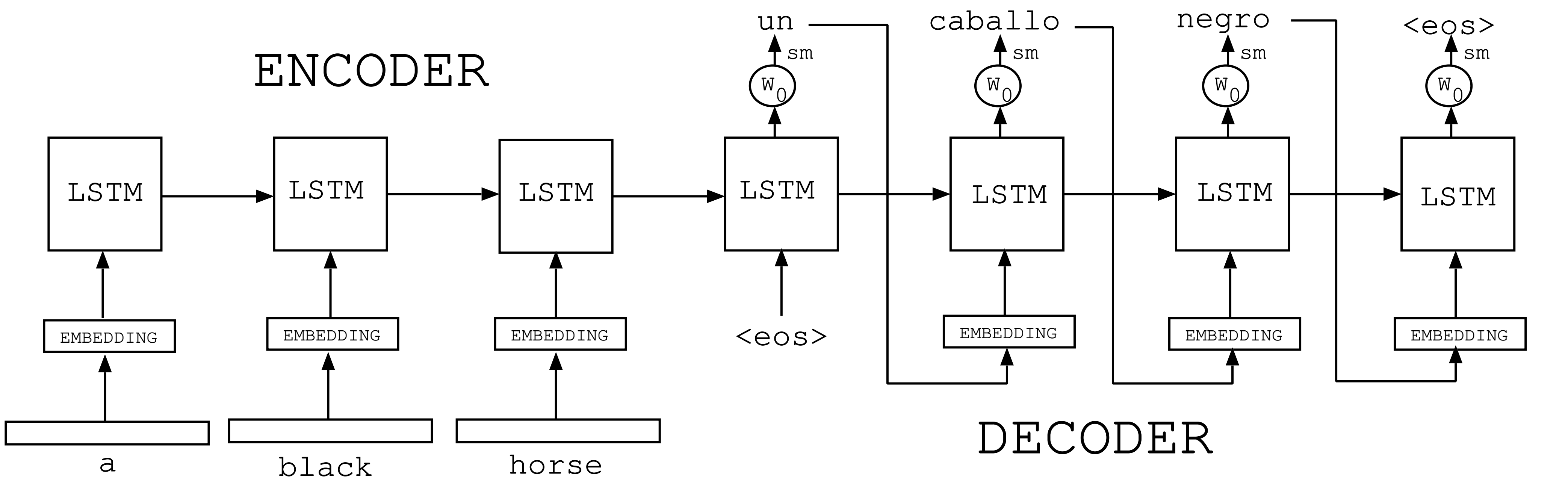
- The RNN-based decoder ($\mathbf{F}_d$) is a kind of target language model

$$\mathbf{h}_0^d = \mathbf{u} \equiv \mathbf{h}_J^e$$

$$\mathbf{h}_i^d = \mathbf{F}_d(\mathbf{W}_E(y_{j-1}), \mathbf{h}_{i-1}^d) = \mathbf{F}_d(\mathbf{y}_{j-1}, \mathbf{h}_{i-1}^d) \quad 1 \leq i \leq I$$

$$p(y_1^I \mid x_1^J) = \prod_{i=1}^{I} p(y_i \mid y_1^{i-1}, u(x_1^J)) = \prod_{i=1}^{I} \mathbf{f}_{sm}(\mathbf{W}_O\ \mathbf{h}_i^d)_{i(y_i)}$$

Text translation using LSTM (Sutskever 2014)



Experiments:
WMT14 English-to-French

Systems	BLEU
Baseline (PB+Neural LM)	33.3
Ensemble of 5 LSTMs	34.8
Reescoring 1000-best with ensemble of 5 LSTMs	36.5
Moses for WMT14	37.0

Index

1 Introduction ▷ 2

2 Syncronized feedforward neural networks ▷ 11

3 Syncronized recurrent neural networks ▷ 14

4 Neural machine translation ▷ 23

○ 5 *Attention models and recurrent neural networks* ▷ 31

6 Attention models for all ▷ 41

7 Training and Decoding with NMT ▷ 67

8 Translation results with neural machine translation ▷ 73

9 Other issues of NMT ▷ 78

10 Bibliography ▷ 90

Encoder-decoder with LSTMs (Vaswani 2017)(Ney 2018)

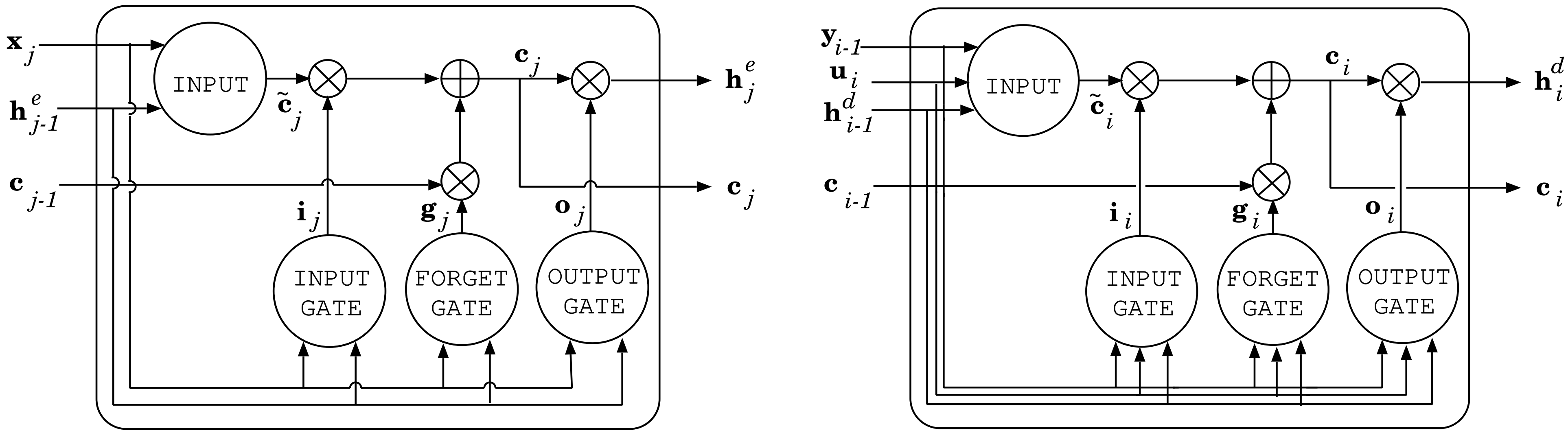
- Goal: Given a source sentence x_1^J and a target sentence y_1^I, compute:

$$p(y_1^I \mid x_1^J) = \prod_{i=1}^{I} p(y_i \mid y_1^{i-1}, u(x_1^J))$$

- Neural networks: LSTMs or GRUs.

- Intra-sentence attention (i, j): For each i, $1 \leq i \leq I$ a probabilistic contribution of j, $1 \leq j \leq J$.

Long Short-Term Memory (LSTM)



$$\mathbf{h}_j^e = \mathbf{F}(\mathbf{x}_j, \mathbf{h}_{j-1}^e) \quad 1 \leq j \leq J \qquad \mathbf{h}_i^d = \mathbf{F}(\mathbf{y}_{i-1}, \mathbf{h}_{i-1}^d, \mathbf{u}_i) \quad 1 \leq i \leq I$$

Attention model (Vaswani 2018)

Given a sequence of encoder states $\mathbf{h}_j^e$ for $1 \leq j \leq J$ and a decoder state $\mathbf{h}_{i-1}^d$ for $1 \leq i \leq I$ an attention model for the decoder state $\mathbf{h}_i^d$ is a function:

$$\mathbf{u}_i = \mathbf{a}(\mathbf{h}_1^e, \ldots, \mathbf{h}_J^e, \mathbf{h}_{i-1}^d)$$

where $\mathbf{a}$ is:

$$\mathbf{u}_i \;=\; \sum_{j=1}^{J} s(\mathbf{h}_j^e, \mathbf{h}_{i-1}^d)\; \mathbf{h}_j^e$$

$s(\mathbf{h}_{i-1}^d, \mathbf{h}_j^e)$ is similarity measure between $\mathbf{h}_{i-1}^d$ and $\mathbf{h}_j^e$, as for example:

$$\mathbf{u}_i \;=\; \sum_{j=1}^{J} f_j(\mathbf{h}_1^e, \ldots, \mathbf{h}_J^e, \mathbf{h}_{i-1}^d)\; \mathbf{h}_j^e$$

f_j is a softmax applied to position j:

$$f_j(\mathbf{h}_1^e, \ldots, \mathbf{h}_J^e, \mathbf{h}_{i-1}^d) = \frac{\exp(\mathbf{h}_{i-1}^d \cdot \mathbf{h}_j^e)}{\sum_{j'=1}^{J} \exp(\mathbf{h}_{i-1}^d \cdot \mathbf{h}_{j'}^e)} \;=\; p(j \mid i) \qquad 1 \leq j \leq J$$

Complete attention model (Vaswani 2018)

Given a sequence of encoder states $\mathbf{h}^e_j$ for $1 \leq j \leq J$ and a decoder state $\mathbf{h}^d_{i-1}$ for $1 \leq i \leq I$ an attention model for the decoder state $\mathbf{h}^d_i$ is a function (V, Q and K stand for "value", "query" and "key", respectively):

$$\mathbf{u}_i = \mathbf{a}(\mathbf{h}^e_1, \ldots, \mathbf{h}^e_J, \mathbf{h}^d_{i-1})$$

where $\mathbf{a}$ is:

$$\mathbf{u}_i = \sum_{j=1}^{J} f_j(\mathbf{h}^e_1, \ldots, \mathbf{h}^e_J, \mathbf{h}^d_{i-1})\ \mathbf{W}_V \mathbf{h}^e_j$$

f_j is a softmax:

$$f_j(\mathbf{h}^e_1, \ldots, \mathbf{h}^e_J, \mathbf{h}^d_{i-1}) = \frac{\exp(\mathbf{W}_Q \mathbf{h}^d_{i-1} \cdot \mathbf{W}_K \mathbf{h}^e_j)}{\sum_{j'=1}^{J} \exp(\mathbf{W}_Q \mathbf{h}^d_{i-1} \cdot \mathbf{W}_K \mathbf{h}^e_{j'})} \qquad 1 \leq j \leq J$$

Using a usual matrix notation: $\mathbf{H}^e = [\mathbf{h}^e_1; \ldots; \mathbf{h}^e_J]$, $\mathbf{H}^d = [\mathbf{h}^d_1; \ldots; \mathbf{h}^d_I]$ and $\mathbf{U} = [\mathbf{u}_1; \ldots; \mathbf{v}_I]$: $\mathbf{U} = \mathbf{A}(\mathbf{W}_Q \mathbf{H}^d, \mathbf{W}_K \mathbf{H}^e, \mathbf{W}_V \mathbf{H}^e)$

NMT with attention model (Bahdanau et al. 2015) (Luong et al. 2015)

- The RNN-based encoder:

$$\mathbf{h}^e_j \;=\; \mathbf{F}_e(\mathbf{W}_E(x_j), \mathbf{h}^e_{j-1}) \;=\; \mathbf{F}_e(\mathbf{x}_j, \mathbf{h}^e_{j-1}) \quad 1 \leq j \leq J \text{ and } \mathbf{h}^e_0 \;=\; \mathbf{0}$$

- The RNN-based decoder with attention model:

$$\mathbf{u}_i = \mathbf{a}(\mathbf{h}^e_1, \ldots, \mathbf{h}^e_J, \mathbf{h}^d_{i-1}) \quad 1 \leq i \leq I$$

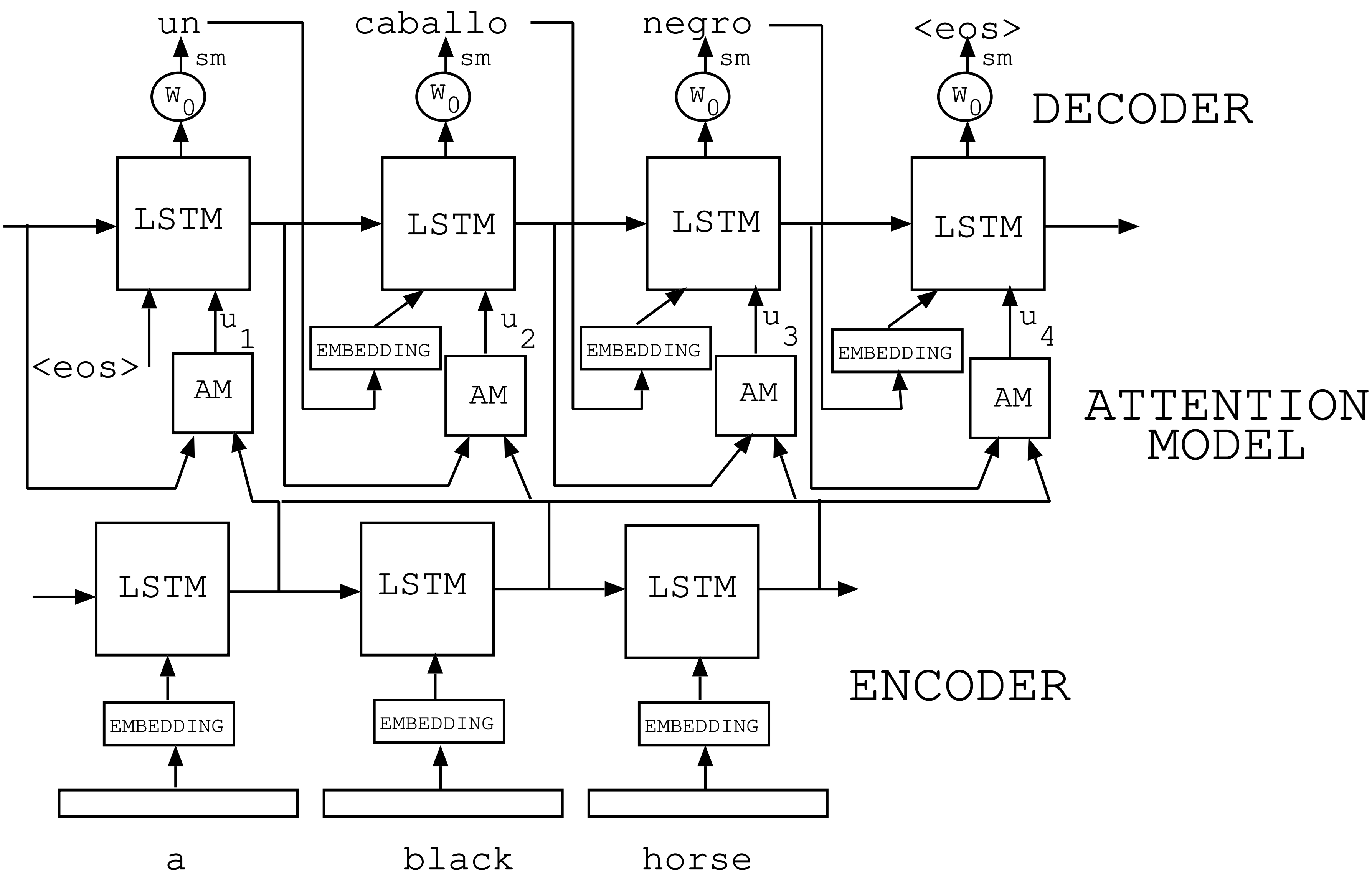
$$\mathbf{h}^d_i \;=\; \mathbf{F}_d(\mathbf{W}_E(y_{i-1}), \mathbf{h}^d_{i-1}, \mathbf{u}_i) \;=\; \mathbf{F}_d(\mathbf{y}_{i-1}, \mathbf{h}^d_{i-1}, \mathbf{u}_i) \quad 1 \leq i \leq I \text{ and } \mathbf{h}^d_0 \;=\; \mathbf{0}$$

A probabilistic distribution over the target dictionary: $\mathbf{f}_{sm}(\mathbf{W}_O\ \mathbf{h}^d_i)$

- Output generation:

$$p(y_1^I \mid x_1^J) = \prod_{i=1}^{I} p(y_i \mid y_1^{i-1}, u(x_1^J)) = \prod_{i=1}^{I} \mathbf{f}_{sm}(\mathbf{W}_O\ \mathbf{h}^d_i)_{i(y_i)}$$

Text translation with attention mechanism (Luong 2015)



Bidirectional RNN for neural machine translation (Bahdanau et al. 2015)

- The RNN-based encoder:

$$\mathbf{h}_j^{e,f} = \mathbf{F}_e(\mathbf{x}_j, \mathbf{h}_{j-1}^{e,f}) \;\; 1 \leq j \leq J \qquad \mathbf{h}_0^{e,f} = \mathbf{0}$$

$$\mathbf{h}_j^{e,b} = \mathbf{F}_e(\mathbf{x}_j, \mathbf{h}_{j+1}^{e,b}) \;\; J \geq j \geq 1 \qquad \mathbf{h}_{J+1}^{e,b} = \mathbf{0}$$

$$\mathbf{h}_j^{e} = [\mathbf{h}_j^{e,f}; \mathbf{h}_j^{e,b}] \;\; 1 \leq j \leq J$$

- The RNN-based decoder with attention model:

$$\mathbf{u}_i = \mathbf{a}(\mathbf{h}_1^e, \ldots, \mathbf{h}_j^e, \mathbf{h}_{i-1}^d) \quad 1 \leq i \leq I$$

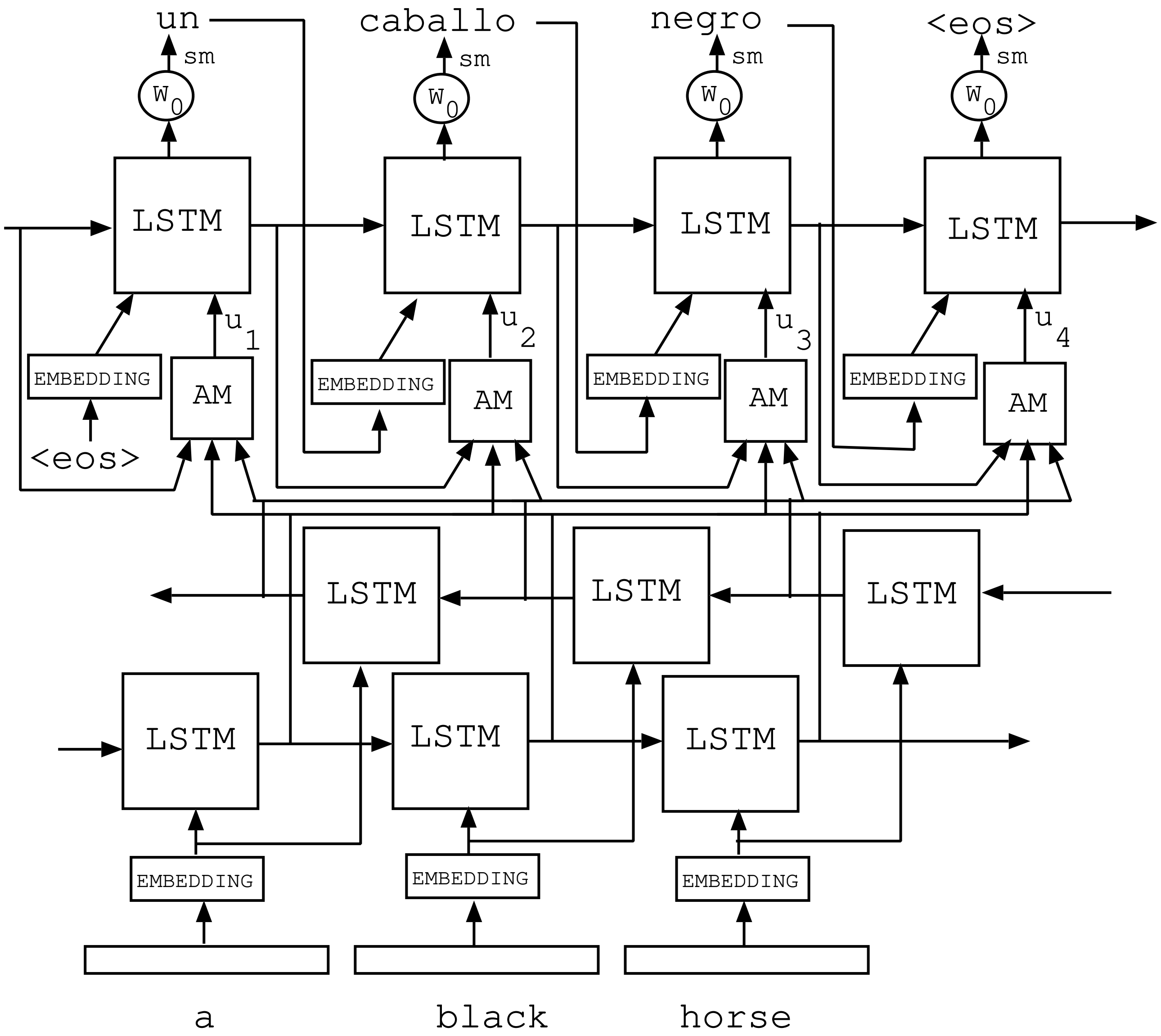
$$\mathbf{h}_i^d = \mathbf{F}_d(\mathbf{y}_{i-1}, \mathbf{h}_{i-1}^d, \mathbf{u}_i) \quad 1 \leq i \leq I \qquad \mathbf{h}_0^d = \mathbf{0}$$

A probabilistic distribution over the target dictionary: $\mathbf{f}_{sm}(\mathbf{W}_O\ \mathbf{h}_i^d)$

- Output generation:

$$p(y_1^I \mid x_1^J) = \prod_{i=1}^{I} p(y_i \mid y_1^{i-1}, u(x_1^J)) = \prod_{i=1}^{I} \mathbf{f}_{sm}(\mathbf{W}_O\ \mathbf{h}_i^d)_{i(y_i)}$$

Text translation with attention mechanism (Luong 2015)



Experiments: WMT14 En-Ge (Luong 2015)

Systems	BLEU
Combination of 8 models	23.0
Best system WMT14	20.7

Experiments in PRHLT: (Peris, 2017)

		TER (%)	
Task	Pair	PB	NMT
Xerox	Es-En	32.5	30.0
EU	Es-En	41.3	45.2

Other encoder-decoder architectures

- Multilayer encoder / multilayer decoder
- Conditional recurrent units (RNN blocks with attention models in between)
- GRUs

Index

1 Introduction ▷ 2

2 Syncronized feedforward neural networks ▷ 11

3 Syncronized recurrent neural networks ▷ 14

4 Neural machine translation ▷ 23

5 Attention models and recurrent neural networks ▷ 31

◦ 6 *Attention models for all* ▷ 41

7 Training and Decoding with NMT ▷ 67

8 Translation results with neural machine translation ▷ 73

9 Other issues of NMT ▷ 78

10 Bibliography ▷ 90

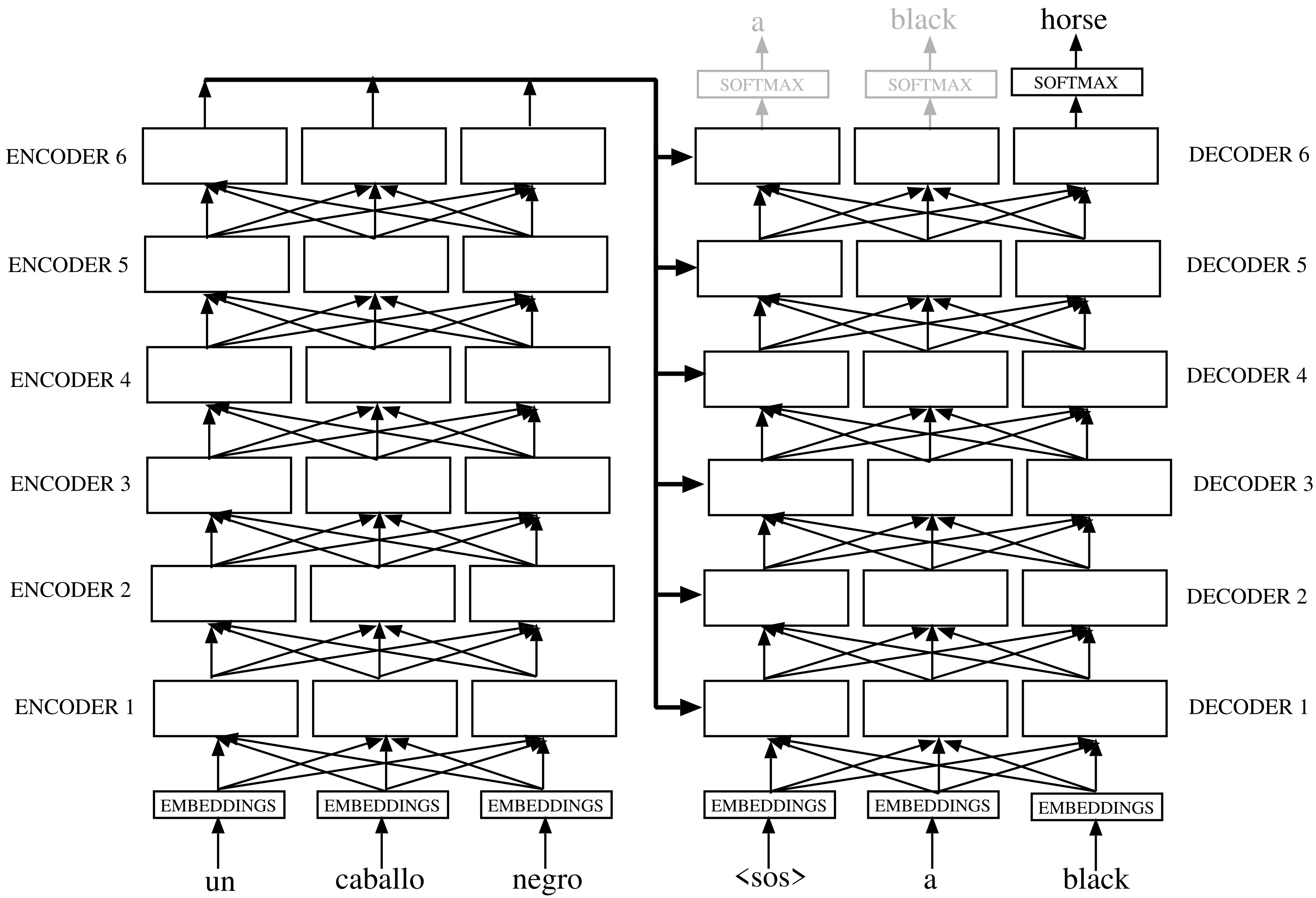
Transformer (Vaswani 2017)(Ney 2018)

- Goal: Given a source sentence x_1^J and a target sentence y_1^I, compute:

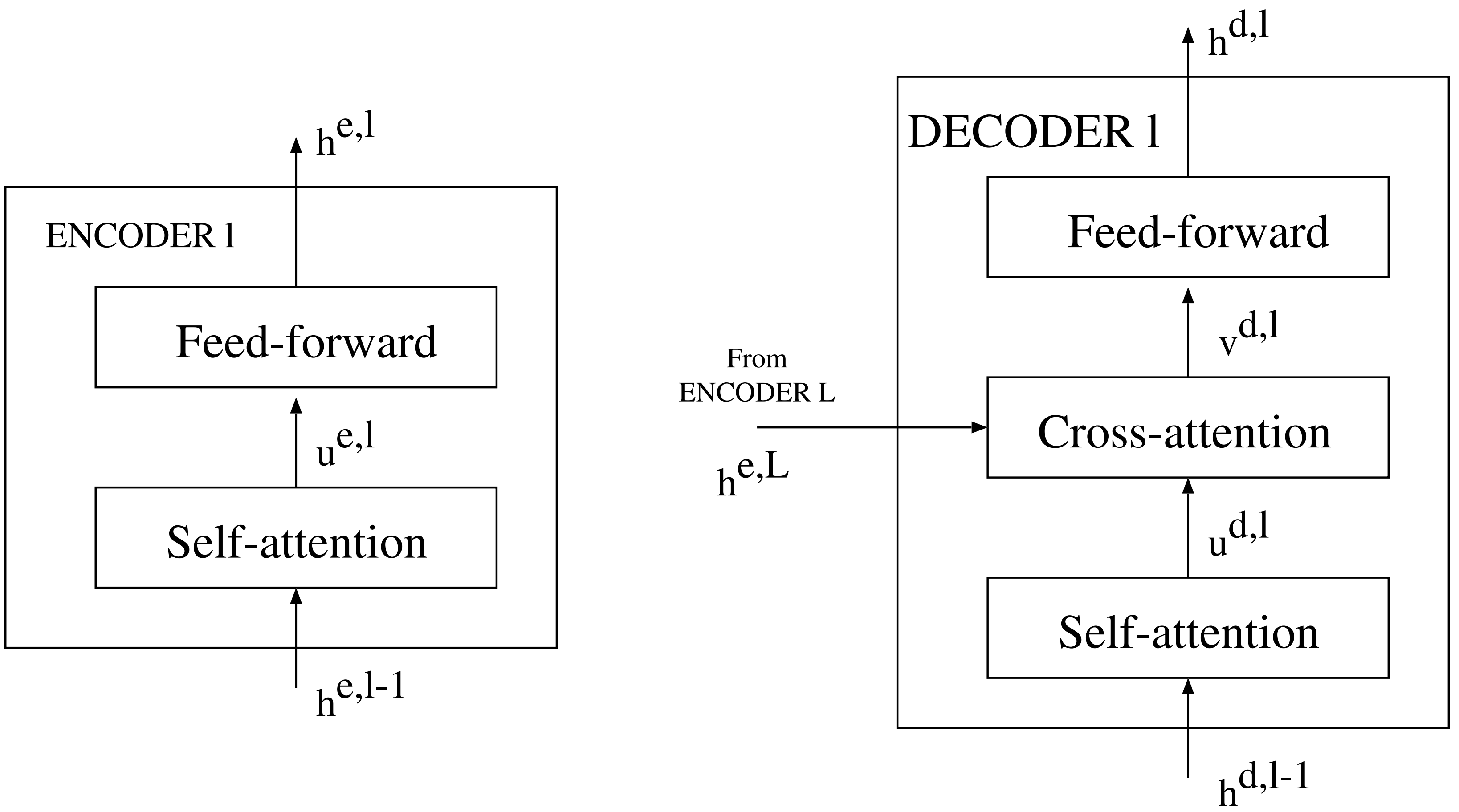
$$p(y_1^I \mid x_1^J) = \prod_{i=1}^{I} p(y_i \mid y_1^{i-1}, u(x_1^J))$$

- Feed-forward networks.
- Self-attention or intra-sentence attention: (j, j') & (i, i') in addition to the cross-attention (i, j).
- Position encoding.
- Multi-head attention.
- Faster than RNN.

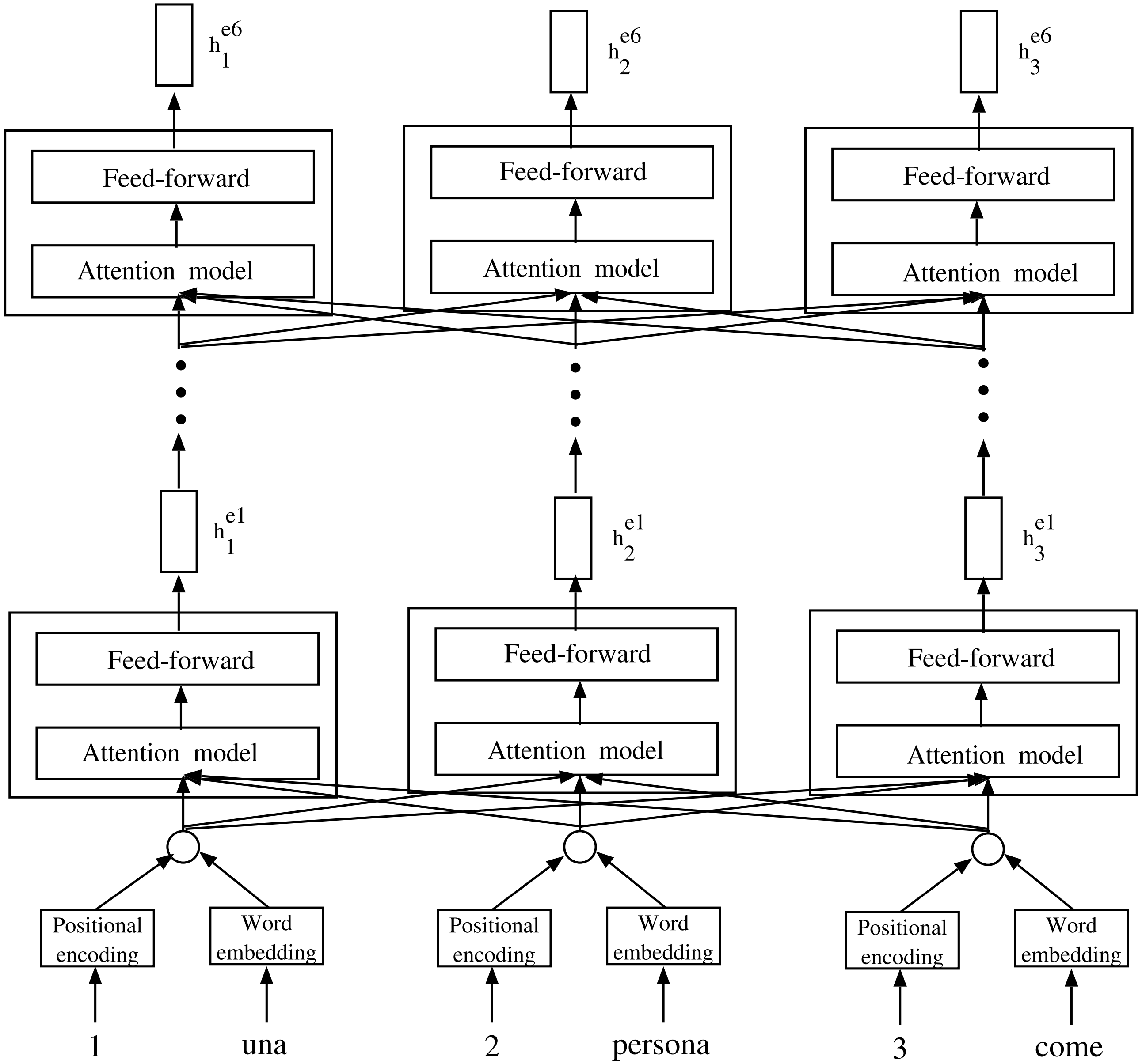
Transformer



Transformer



A simplified view of the encoder in the Transformer



Positional and word encodings

- A word x from a ordered vocabulary V_X with index $i(x)$
- Word embeddings: $\mathbf{W_E}(x) \equiv [\mathbf{W_E}]_{i(x)} = \mathbf{x} \in \mathbb{R}^{D_W}$: a row of $\mathbf{W_E}$ is the word embedding of x
- Positional embeddings: Given a sentence x_1^J the positional embedding of position j, $1 \leq j \leq J$ is $\mathbf{p}_j \in \mathbb{R}^{D_P}$:

$$\begin{aligned} p_{j,2k} &= \sin(j/10000^{2k/D_P}) \\ p_{j,2k+1} &= \cos(j/10000^{2k/D_P}) \end{aligned}$$

- Word plus positional embeddings: $\mathbf{h}_j^{e,0} = \mathbf{x}_j + \mathbf{p}_j$ for $1 \leq j \leq J$
- Relative positional embeddings.
- Learned positional embeddings.

The encoder of Transformer

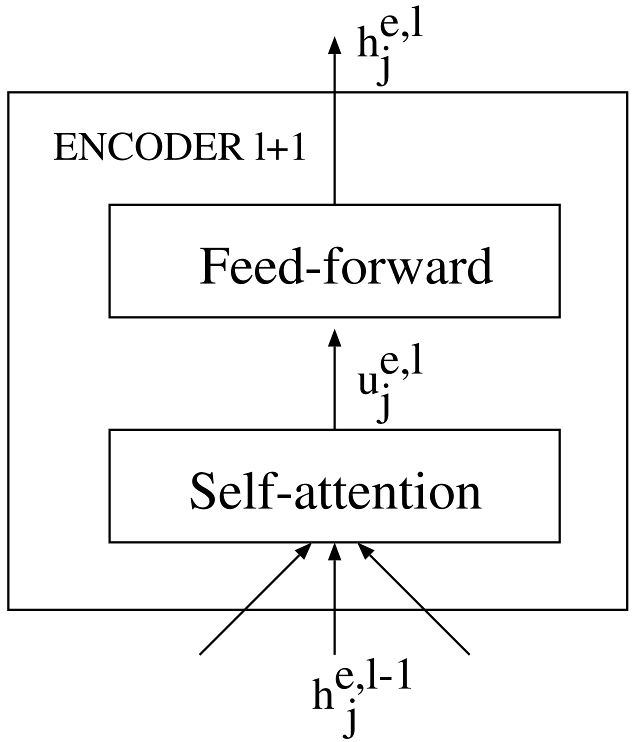
Given a source sentence x_1^J,

- Initialization: $\mathbf{h}_j^{e,0} = \mathbf{x}_j + \mathbf{p}_j \;\; (= \mathcal{E}(x_j) + \mathcal{E}(j)) \quad 1 \leq j \leq J$
- In layer l of the encoder($1 \leq l \leq L$)
  - Self-attention model:

$$\mathbf{u}_j^{e,l} = \mathbf{a}(\mathbf{h}_1^{e,l-1}, \ldots, \mathbf{h}_J^{e,l-1}, \mathbf{h}_j^{e,l-1}) \quad 1 \leq i \leq J$$

  - Feed-forward network:

$$\mathbf{h}_j^{e,l} \;=\; \mathbf{F}_f(\mathbf{u}_j^{e,l}) \quad 1 \leq j \leq J$$



Self-attention models for the encoder

Given a sequence of encoder states $\mathbf{h}_j^{e,l-1}$ for $1 \leq j \leq J$ in layer $1 \leq l \leq L$, a self-attention model for a position j in a layer l is a function:

$$\mathbf{u}_j^{e,l} = \mathbf{a}(\mathbf{h}_1^{e,l-1}, \ldots, \mathbf{h}_J^{e,l-1}, \mathbf{h}_j^{e,l-1}) \quad 1 \leq i \leq J$$

where $\mathbf{a}$ is:

$$\mathbf{u}_j^{e,l} = \sum_{j'=1}^{J} f_{j'}(\mathbf{h}_1^{e,l-1}, \ldots, \mathbf{h}_J^{e,l-1}, \mathbf{h}_j^{e,l-1})\ \mathbf{h}_{j'}^{e,l-1} \quad 1 \leq i \leq J$$

$$f_{j'}(\mathbf{h}_1^{e,l-1}, \ldots, \mathbf{h}_J^{e,l-1}, \mathbf{h}_j^{e,l-1}) = \frac{\exp(\mathbf{h}_j^{e,l-1} \cdot \mathbf{h}_{j'}^{e,l-1})}{\sum_{j"=1}^{J} \exp(\mathbf{h}_j^{e,l-1} \cdot \mathbf{h}_{j"}^{e,l-1})} = p(j' \mid j) \quad 1 \leq j, j' \leq J$$

Complete self-attention models for the encoder

Given a sequence of encoder states $\mathbf{h}_j^{e,l-1}$ for $1 \leq j \leq J$ in layer $1 \leq l \leq L$, a self-attention model for a position j in a layer l is a function:

$$\mathbf{u}_j^{e,l} = \mathbf{a}(\mathbf{h}_1^{e,l-1}, \ldots, \mathbf{h}_J^{e,l-1}, \mathbf{h}_j^{e,l-1}) \quad 1 \leq i \leq J$$

where $\mathbf{a}$ is:

$$\mathbf{u}_j^{e,l} = \sum_{j'=1}^{J} f_{j'}(\mathbf{h}_1^{e,l-1}, \ldots, \mathbf{h}_J^{e,l-1}, \mathbf{h}_j^{e,l-1})\, \mathbf{W}_V^{e,l} \mathbf{h}_{j'}^{e,l-1} \quad 1 \leq i \leq J$$

$$f_{j'}(\mathbf{h}_1^{e,l-1}, \ldots, \mathbf{h}_J^{e,l-1}, \mathbf{h}_j^{e,l-1}) = \frac{\exp(\mathbf{W}_Q^{e,l}\mathbf{h}_j^{e,l-1} \cdot \mathbf{W}_K^{e,l}\mathbf{h}_{j'}^{e,l-1})}{\sum_{j"=1}^{J} \exp(\mathbf{W}_Q^{e,l}\mathbf{h}_j^{e,l-1} \cdot \mathbf{W}_K^{e,l}\mathbf{h}_{j"}^{e,l-1})} \quad 1 \leq j, j' \leq J$$

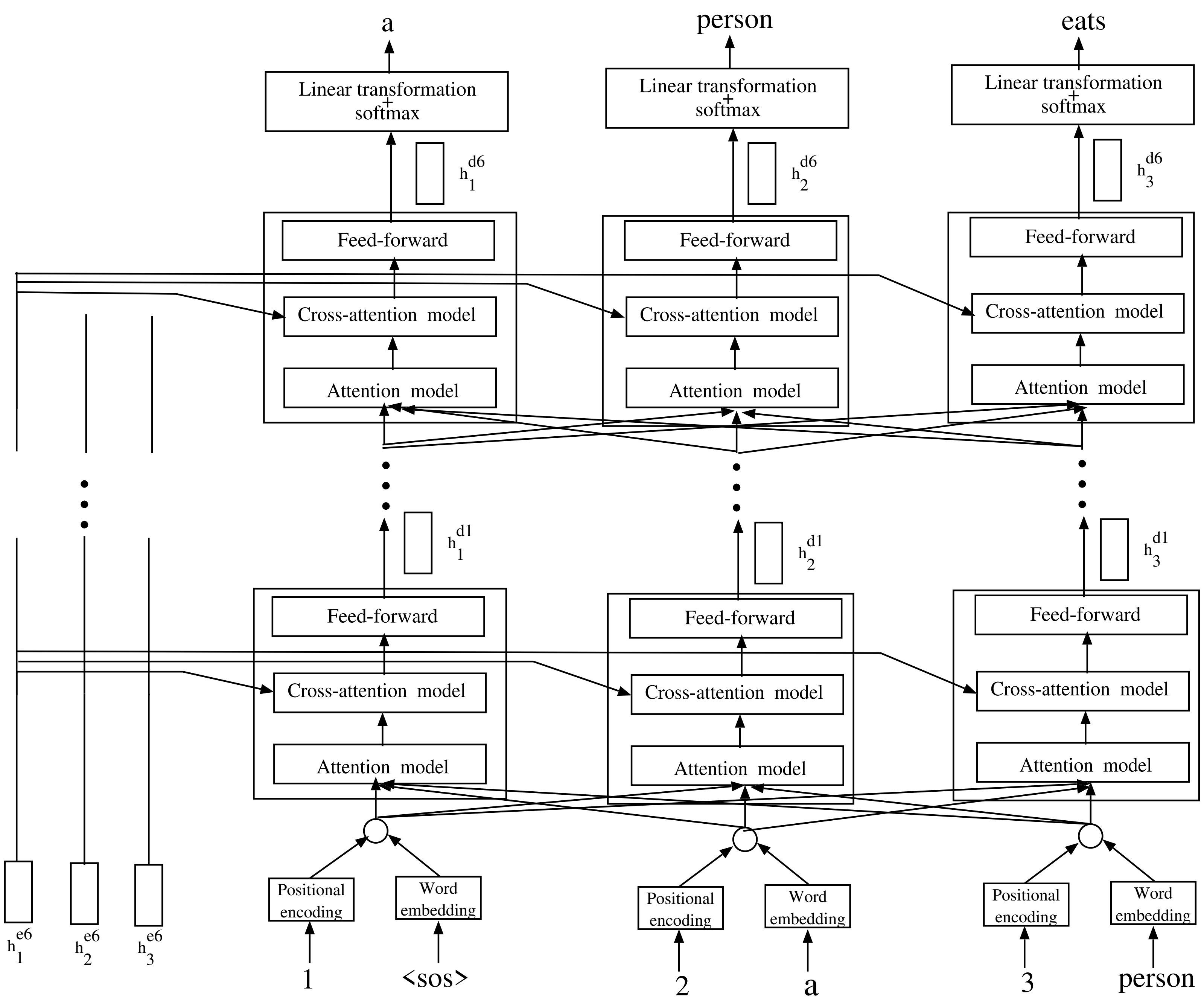
Feed-forward network

- Feed-forward networks, $\mathbf{F}_f$ for layer l

$$\mathbf{F}_f(\mathbf{u}_j^{e,l}) \;=\; \mathbf{W}_2^{e,l}\,\mathbf{f}_{ReLU}(\mathbf{W}_1^{e,l}\,\mathbf{u}_j^{e,l} + \mathbf{b}_1^{e,l}) + \mathbf{b}_2^{e,l}$$

where $\mathbf{W}_2^{e,l}$, $\mathbf{W}_1^{e,l}$ are the weights of the second and first layer of the feed-forward networks, and $\mathbf{b}_2^{e,l}$ and $\mathbf{b}_1^{e,l}$ the corresponding bias.

A simplified view of the decoder in the Transformer



The decoder of Transformer

Given a sequence of states of the encoder $\mathbf{h}_j^{e,L} \quad 1 \leq j \leq J$,

- For $1 \leq i \leq I$
  - Initialization: $\mathbf{u}_{i-1}^{d,0} = \mathbf{y}_{i-1} = \mathcal{E}(y_{i-1})$, ($y_0 = " < sos > "$)
  - In layer l of the decoder ($1 \leq l \leq L$):
    - ∗ Self-attention model:

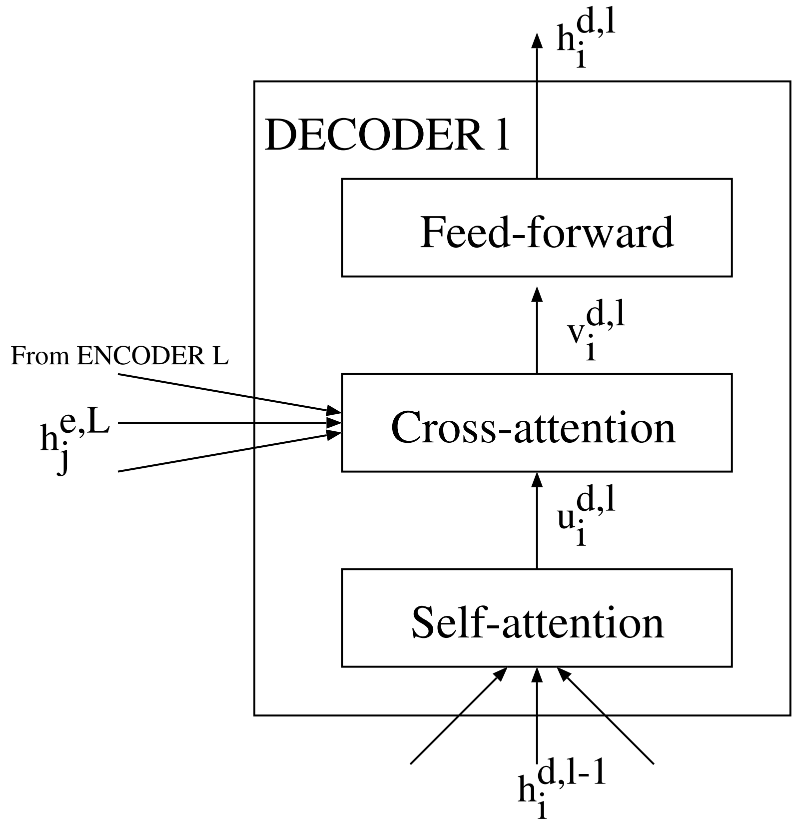
$$\mathbf{u}_i^{d,l} = \mathbf{a}(\mathbf{h}_1^{d,l-1}, \ldots, \mathbf{h}_i^{d,l-1}, \mathbf{h}_i^{d,l-1})$$

    - ∗ Cross-attention model:

$$\mathbf{v}_i^{d,l} = \mathbf{a}(\mathbf{h}_1^{e,L}, \ldots, \mathbf{h}_J^{e,L}, \mathbf{u}_i^{d,l})$$

    - ∗ Feed-forward network:

$$\mathbf{h}_i^{d,l} = \mathbf{F}_f(\mathbf{v}_i^{d,l})$$



Self-attention models for the decoder

Given a sequence of decoder states $\mathbf{h}_i^{d,l-1}$ for $1 \leq i \leq I$, a self-attention model for a position i in a layer l is a function:

$$\mathbf{u}_i^{d,l} = \mathbf{a}(\mathbf{h}_1^{d,l-1}, \ldots, \mathbf{h}_I^{d,l-1}, \mathbf{h}_i^{d,l-1}) \quad 1 \leq i \leq I$$

where $\mathbf{a}$ is:

$$\mathbf{u}_i^{d,l} = \sum_{i'=1}^{I} f_{i'}(\mathbf{h}_1^{d,l-1}, \ldots, \mathbf{h}_I^{d,l-1}, \mathbf{h}_i^{d,l-1})\ \mathbf{h}_{i'}^{d,l-1} \quad 1 \leq i \leq I$$

$$f_{i'}(\mathbf{h}_1^{d,l-1}, \ldots, \mathbf{h}_I^{d,l-1}, \mathbf{h}_i^{d,l-1}) = \frac{\exp(\mathbf{h}_i^{d,l-1} \cdot \mathbf{h}_{i'}^{d,l-1})}{\sum_{i"=1}^{I} \exp(\mathbf{h}_i^{d,l-1} \cdot \mathbf{h}_{i"}^{d,l-1})} = p(i' \mid i) \quad 1 \leq i, i' \leq I$$

Complete self-attention models for the decoder

Given a sequence of decoder states $\mathbf{h}_i^{d,l-1}$ for $1 \le i \le I$, a self-attention model for a position i in a layer l is a function:

$$\mathbf{u}_i^{d,l} = \mathbf{a}(\mathbf{h}_1^{d,l-1}, \ldots, \mathbf{h}_I^{d,l-1}, \mathbf{h}_i^{d,l-1}) \quad 1 \le i \le I$$

where $\mathbf{a}$ is:

$$\mathbf{u}_i^{d,l} = \sum_{i'=1}^{I} f_{i'}(\mathbf{h}_1^{d,l-1}, \ldots, \mathbf{h}_I^{d,l-1}, \mathbf{h}_i^{d,l-1}) \, \mathbf{W}_V^{d,l} \mathbf{h}_{i'}^{d,l-1} \quad 1 \le i \le I$$

$$f_{i'}(\mathbf{h}_1^{d,l-1}, \ldots, \mathbf{h}_I^{d,l-1}, \mathbf{h}_i^{d,l-1}) = \frac{\exp(\mathbf{W}_Q^{d,l}\mathbf{h}_i^{d,l-1} \cdot \mathbf{W}_K^{d,l}\mathbf{h}_{i'}^{d,l-1})}{\displaystyle\sum_{i''=1}^{I} \exp(\mathbf{W}_Q^{d,l}\mathbf{h}_i^{d,l-1} \cdot \mathbf{W}_K^{d,l}\mathbf{h}_{i''}^{d,l-1})} \quad 1 \le i, i' \le I$$

Cross-attention models for the decoder

Given a sequence of output of self-attentions in the decoder $\mathbf{u}_{i'}^{d,l-1}$ for $1 \leq i' \leq i$ and some i, a cross-attention model for a position $i' \leq i$ in a layer l is a function:

$$\mathbf{v}_{i'}^{d,l} = \mathbf{a}(\mathbf{h}_1^{e,L}, \dots, \mathbf{h}_J^{1,L}, \mathbf{u}_{i'}^{d,l}) \quad 1 \leq i' \leq i$$

where $\mathbf{a}_c$ is in:

$$\mathbf{v}_{i'}^{d,l} = \sum_{j=1}^{J} f_j(\mathbf{h}_1^{e,L}, \dots, \mathbf{h}_J^{e,L}, \mathbf{u}_{i'}^{d,l})\ \mathbf{h}_j^{e,L}$$

$$f_j(\mathbf{h}_1^{e,L}, \dots, \mathbf{h}_J^{e,L}, \mathbf{u}_{i'}^{d,l}) = \frac{\exp(\mathbf{u}_{i'}^{d,l} \cdot \mathbf{h}_j^{e,L})}{\sum_{j=1}^{J} \exp(\mathbf{u}_{i'}^{d,l} \cdot \mathbf{h}_j^{e,L})} = p(j \mid i'); 1 \leq i' \leq i, \quad 1 \leq j \leq J$$

Complete cross-attention models for the decoder

Given a sequence of output of self-attentions in the decoder $\mathbf{u}_{i'}^{d,l-1}$ for $1 \leq i' \leq i$ and some i, a cross-attention model for a position $i' \leq i$ in a layer l is a function:

$$\mathbf{v}_{i'}^{d,l} = \mathbf{a}(\mathbf{h}_1^{e,L}, \ldots, \mathbf{h}_J^{1,L}, \mathbf{u}_{i'}^{d,l}) \quad 1 \leq i' \leq i$$

where $\mathbf{a}_c$ is in:

$$\mathbf{v}_{i'}^{d,l} = \sum_{j=1}^{J} f_j(\mathbf{h}_1^{e,L}, \ldots, \mathbf{h}_J^{e,L}, \mathbf{u}_{i'}^{d,l})\ \mathbf{W}_V^{c,l} \mathbf{h}_j^{e,L}$$

$$f_j(\mathbf{h}_1^{e,L}, \ldots, \mathbf{h}_J^{e,L}, \mathbf{u}_{i'}^{d,l}) = \frac{\exp(\mathbf{W}_Q^{c,l}\mathbf{u}_{i'}^{d,l} \cdot \mathbf{W}_K^{c,l}\mathbf{h}_j^{e,L})}{\sum_{j=1}^{J} \exp(\mathbf{W}_Q^{c,l}\mathbf{u}_{i'}^{d,l} \cdot \mathbf{W}_K^{c,l}\mathbf{h}_j^{e,L})} \quad 1 \leq i' \leq i, \quad 1 \leq j \leq J$$

Feed-forward network

- Feed-forward networks, $\mathbf{F}_f$: Given a $\mathbf{v}_i^{d,l} \in \mathbb{R}^d$.

$$\mathbf{F}_f(\mathbf{v}_i^{d,l}) \;=\; \mathbf{W}_2^{d.l}\, \mathbf{f}_{ReLU}(\mathbf{W}_1^{d.l}\, \mathbf{v}_i^{d,l} + \mathbf{b}_1^{d.l}) + \mathbf{b}_2^{d.l}$$

where $\mathbf{W}_2^{d.l}$, $\mathbf{W}_1^{d.l}$ are the weights of the second and first layer of the feed-forward networks, and $\mathbf{b}_2^{d.l}$ and $\mathbf{b}_1^{d.l}$ the corresponding bias.

The decoder of Transformer (Vaswani 2017)

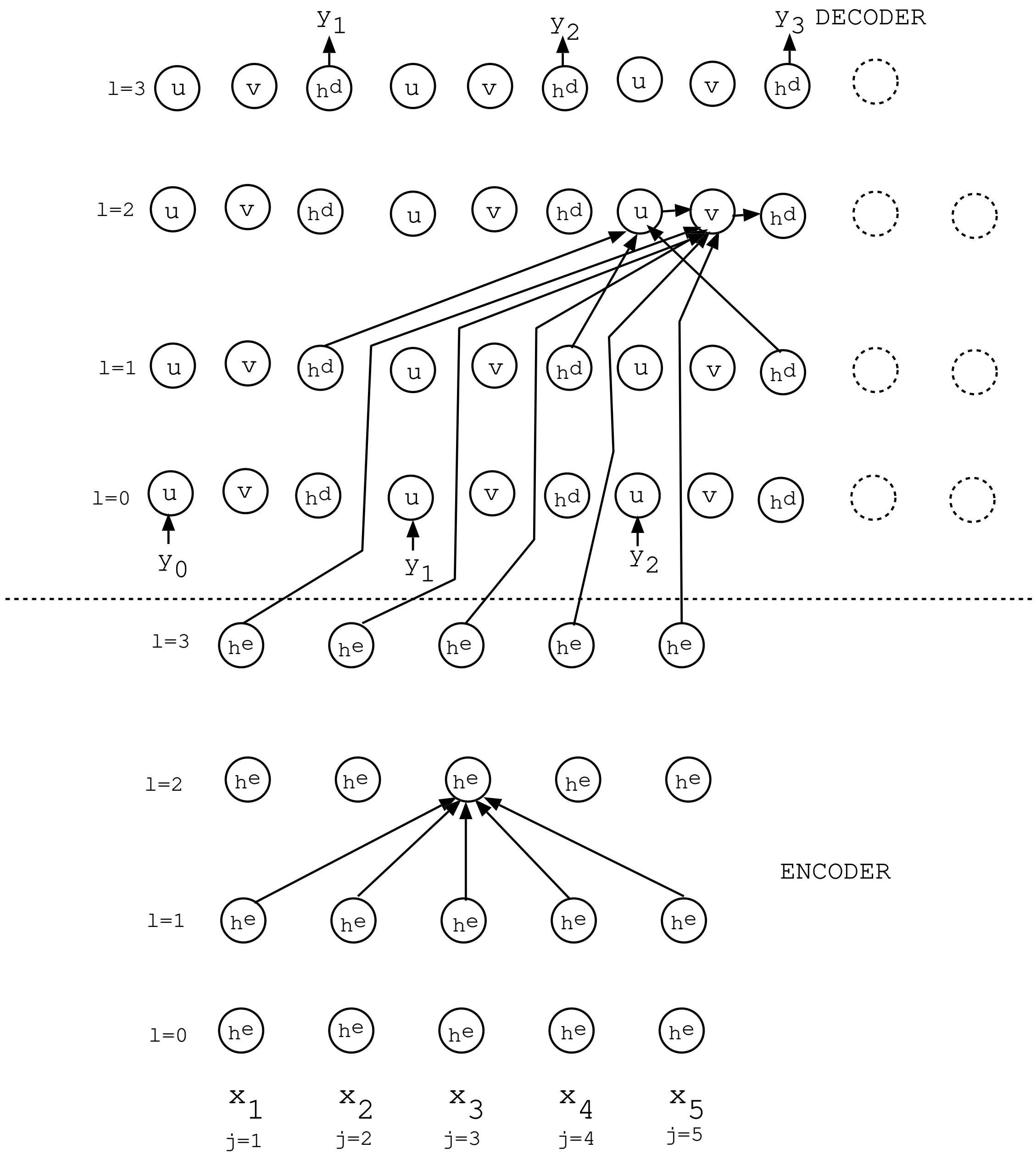
- For each the state of the decoder in the last layer $\mathbf{h}_i^{d,L} \quad 1 \leq i \leq I$:

  A probabilistic distribution over the target dictionary: $\mathbf{f}_{sm}(\mathbf{W}_O \; \mathbf{h}_i^{d,L})$

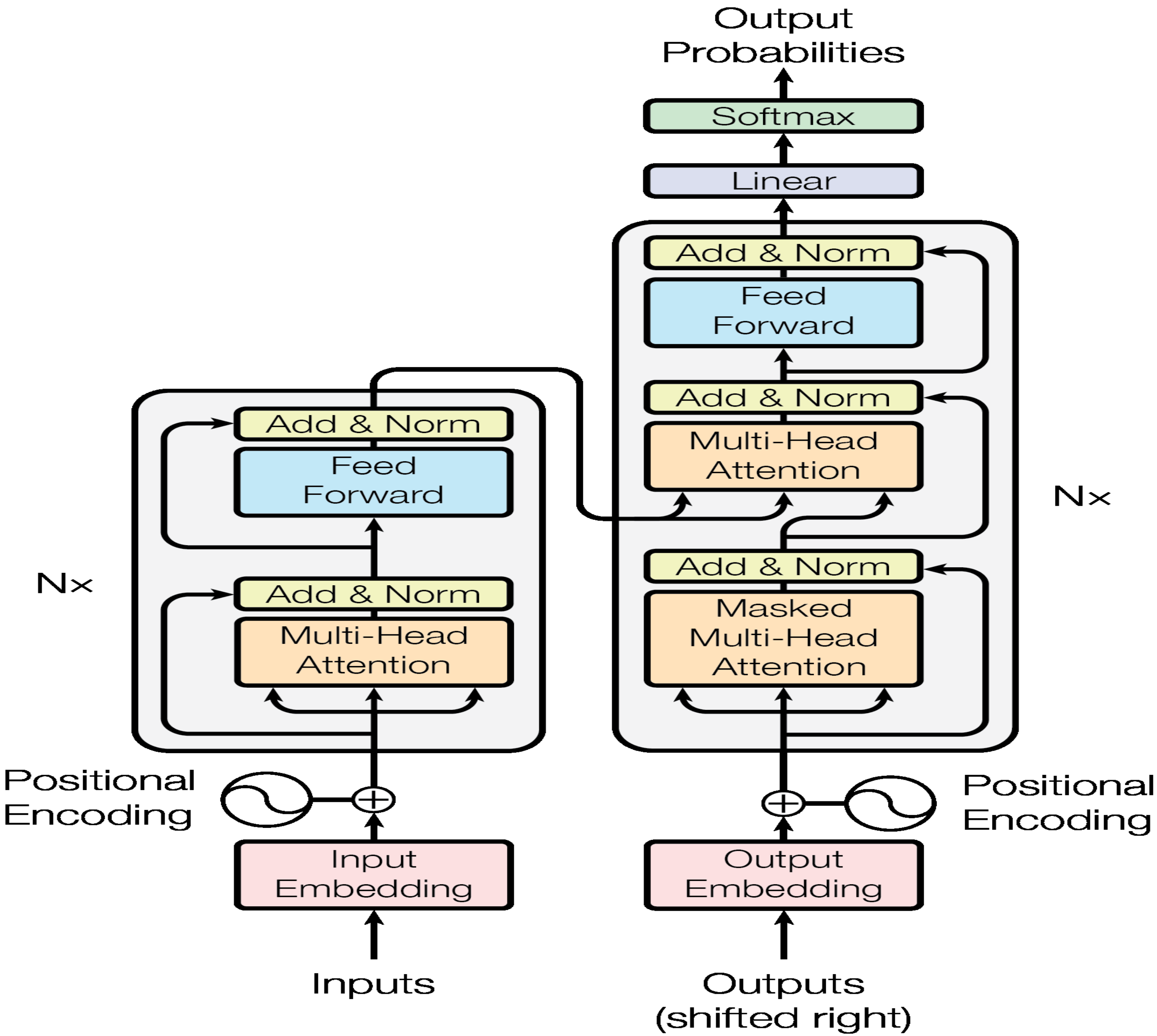
- Output generation:

$$p(y_1^I \mid x_1^J) = \prod_{i=1}^{I} p(y_i \mid y_1^{i-1}, u(x_1^J)) = \prod_{i=1}^{I} \mathbf{f}_{sm}(\mathbf{W} \; \mathbf{h}_i^{d,L})_{i(y_i)}$$

The whole process (Ney 2018)



Transformer (Vaswani 2017)



Some details and additional features of Transformer

- Residual networks
- Layer normalization
- Multi-head attention

Some details and additional features of Transformer (Xiong arXiv 2020)

- Given $\mathbf{z} \in \mathbb{R}^d$ and a function $\mathbf{f} : \mathbb{R}^d \rightarrow \mathbb{R}^d$, a residual function $\mathbf{R}$ is defined as:

$$\mathbf{R}(\mathbf{z}, \mathbf{f}(\mathbf{z})) = \mathbf{f}(\mathbf{z}) + \mathbf{z}$$

- Given a sequence of vectors $\mathbf{z}_1, \ldots, \mathbf{z}_K$ from a layer with $\mathbf{z}_k \in \mathbb{R}^d$ for $1 \leq k \leq K$, a layer normalization $\mathbf{N}$ is $\mathbf{N}(\mathbf{z}_1, \ldots, \mathbf{z}_K) = (\bar{\mathbf{z}}_1, \ldots, \bar{\mathbf{z}}_K)$ such that:

$$\bar{z}_{k,i} = \gamma \, \frac{z_{k,i} - \mu_i}{\sigma_i} + \beta \quad 1 \leq k \leq K, \; 1 \leq i \leq d$$

where γ and β are hyper-parameters:

$$\mu_i = \frac{\sum_{k=1}^{K} z_{k,i}}{K} \quad \text{and} \quad \sigma_i^2 = \frac{\sum_{k=1}^{K} (z_{k,i} - \mu_i)^2}{K} \quad 1 \leq i \leq d$$

Compact notation: Matrix representation

- $LN \equiv$ layer normalization, $\mathbf{F} \equiv$ feed-forward network and $\mathbf{A} \equiv$ attention mechanism.

- Encoder: $\mathbf{H}^{e,l} = [\mathbf{h}_1^{e,l}; \ldots; \mathbf{h}_J^{e,l}]$ for $1 \leq l \leq L$

$$\begin{aligned}
\mathbf{U}^{e,l} &= LN(\mathbf{A}(\mathbf{W}_Q^{e,l}\mathbf{H}^{e,l-1}, \mathbf{W}_K^{e,l}\mathbf{H}^{e,l-1}, \mathbf{W}_V^{e,l}\mathbf{H}^{e,l-1}) + \mathbf{H}^{e,l-1}) \\
\mathbf{H}^{e,l} &= LN(\mathbf{F}(\mathbf{U}^{e,l}) + \mathbf{U}^{e,l})
\end{aligned}$$

- Decoder: $\mathbf{H}^{d,l} = [\mathbf{h}_1^{d,l}; \ldots; \mathbf{h}_J^{d,l}]$ for $1 \leq l \leq L$

$$\begin{aligned}
\mathbf{U}^{d,l} &= LN(\mathbf{A}(\mathbf{W}_Q^{d,l}\mathbf{H}^{d,l-1}, \mathbf{W}_K^{d,l}\mathbf{H}^{d,l-1}, \mathbf{W}_V^{d,l}\mathbf{H}^{d,l-1}) + \mathbf{H}^{d,l-1}) \\
\mathbf{C}^{d,l} &= LN(\mathbf{A}(\mathbf{W}_Q^{d,l}\mathbf{U}^{d,l-1}, \mathbf{W}_K^{d,l}\mathbf{H}^{e,L}, \mathbf{W}_V^{d,l}\mathbf{H}^{e,L}) + \mathbf{U}^{d,l}) \\
\mathbf{H}^{d,l} &= LN(\mathbf{F}(\mathbf{C}^{d,l}) + \mathbf{C}^{d,l}))
\end{aligned}$$

Multi-head attention in Transformer

For N heads and $1 \leq l \leq L$, $1 \leq j \leq J$ and $1 \leq i \leq I$:

- Encoder self-attention:

$$\mathbf{u}_{j,n}^{e,l} = \mathbf{a}_n(\mathbf{h}_1^{e,l-1}, \ldots, \mathbf{h}_J^{e,l-1}, \mathbf{h}_j^{e,l-1}) \quad 1 \leq j \leq J \text{ and } 1 \leq n \leq N$$

$$= \sum_{j'=1}^{J} \frac{\exp(\mathbf{W}_Q^{e,l,n}\mathbf{h}_j^{e,l-1} \cdot \mathbf{W}_K^{e,l,n}\mathbf{h}_{j'}^{e,l-1})}{\sum_{j"=1}^{J} \exp(\mathbf{W}_Q^{e,l,n}\mathbf{h}_j^{e,l-1} \cdot \mathbf{W}_K^{e,l,n}\mathbf{h}_{j"}^{e,l-1})} \mathbf{W}_V^{e,l,n}\mathbf{h}_{j'}^{e,l-1} \qquad 1 \leq j, j' \leq J$$

$\mathbf{W}_Q^{e,l,n}, \mathbf{W}_K^{e,l,n}, \mathbf{W}_V^{e,l,n}$ are matrix of $D_W/N \times D_W$

$$\mathbf{u}_j^{e,l} = \left[\mathbf{u}_{j,1}^{e,l}, \ldots, \mathbf{u}_{j,N}^{e,l}\right]; \quad \mathbf{h}_j^{e,l} = \mathbf{F}_f(\mathbf{u}_j^{e,l}) \quad 1 \leq j \leq J$$

Multi-head attention in Transformer

For N heads and $1 \leq l \leq L$, $1 \leq j \leq J$ and $1 \leq i \leq I$:

- Decoder self-attention:

$$\begin{aligned}
\mathbf{u}_{i,n}^{d,l} &= \mathbf{a}_n(\mathbf{h}_1^{d,l-1}, \ldots, \mathbf{h}_J^{d,l-1},\ \mathbf{h}_i^{d,l-1}) \quad 1 \leq i \leq I \text{ and } 1 \leq n \leq N \\
&= \sum_{i'=1}^{I} \frac{\exp(\mathbf{W}_Q^{d,l,n}\mathbf{h}_i^{d,l-1} \cdot \mathbf{W}_K^{d,l,n}\mathbf{h}_{i'}^{d,l-1})}{\sum_{i"=1}^{I}\exp(\mathbf{W}_Q^{d,l,n}\mathbf{h}_i^{d,l-1} \cdot \mathbf{W}_K^{d,l,n}\mathbf{h}_{i"}^{d,l-1})}\mathbf{W}_V^{d,l,n}\mathbf{h}_{i'}^{d,l-1} \qquad 1 \leq i, i' \leq I
\end{aligned}$$

$\mathbf{W}_Q^{d,l,n}, \mathbf{W}_K^{d,l,n}, \mathbf{W}_V^{d,l,n}$ are matrix of $D_W/N \times D_W$

$$\mathbf{u}_i^{d,l} = \left[\mathbf{u}_{i,1}^{d,l}, \ldots, \mathbf{u}_{i,N}^{d,l}\right]; \quad \mathbf{h}_i^{d,l} = \mathbf{F}_f(\mathbf{u}_i^{d,l}) \quad 1 \leq i \leq I$$

Multi-head attention in Transformer

For N heads and $1 \leq l \leq L$, $1 \leq j \leq J$ and $1 \leq i \leq I$:

- Decoder cross-attention:

$$
\begin{aligned}
\mathbf{v}_{i,n}^{d,l} &= \mathbf{a}_n(\mathbf{h}_1^{e,L}, \ldots, \mathbf{h}_J^{e,L},\ \mathbf{u}_i^{d,l}) \quad 1 \leq i \leq I \text{ and } 1 \leq n \leq N \\
&= \sum_{j=1}^{J} \frac{\exp(\mathbf{W}_Q^{c,l,n}\mathbf{u}_i^{d,l} \cdot \mathbf{W}_K^{c,l,n}\mathbf{h}_j^{e,L})}{\sum_{j=1}^{J} \exp(\mathbf{W}_Q^{c,l,n}\mathbf{u}_i^{d,l} \cdot \mathbf{W}_K^{c,l,n}\mathbf{h}_j^{e,L})} \mathbf{W}_V^{c,n}\mathbf{h}_j^{e,L} \qquad 1 \leq i \leq I
\end{aligned}
$$

$\mathbf{W}_Q^{c,l,n}, \mathbf{W}_K^{c,l,n}, \mathbf{W}_V^{c,l,n}$ are matrix of $D_W/N \times D_W$

$$
\mathbf{v}_i^{d,l} = \left[\mathbf{v}_{i,1}^{d,l}, \ldots, \mathbf{v}_{i,N}^{d,l}\right]; \quad \mathbf{h}_i^{d,l} = \mathbf{F}_f(\mathbf{v}_i^{d,l}) \quad 1 \leq i \leq I
$$

Index

1 Introduction ▷ 2

2 Syncronized feedforward neural networks ▷ 11

3 Syncronized recurrent neural networks ▷ 14

4 Neural machine translation ▷ 23

5 Attention models and recurrent neural networks ▷ 31

6 Attention models for all ▷ 41

◦ 7 *Training and Decoding with NMT* ▷ 67

8 Translation results with neural machine translation ▷ 73

9 Other issues of NMT ▷ 78

10 Bibliography ▷ 90

Training with NMT (Koehn, 2017)

- Given a training set of bilingual pairs T, maximize the log-likelihood:

$$
\begin{aligned}
\widehat{\mathbf{W}} &= \underset{\mathbf{W}}{\text{argmax}}\, \mathcal{F}_T(\mathbf{W}) \equiv \underset{\mathbf{W}}{\text{argmax}} \sum_{(x_1^J, y_1^I) \in T} \log p(y_1^I \mid x_1^J; \mathbf{W}) \\
&= \underset{\mathbf{W}}{\text{argmax}} \sum_{(x_1^J, y_1^I) \in T} \sum_{i=1}^{I} \log p(y_i \mid y_1^{i-1}, x_1^J; \mathbf{W}) \\
&= \underset{\mathbf{W}}{\text{argmax}} \sum_{(x_1^J, y_1^I) \in T} \sum_{i=1}^{I} \log \mathbf{f}_{sm}(\mathbf{W}_O\, \mathbf{h}_i^d(x_1^J))_{i(y_i)}
\end{aligned}
$$

- Optimization algorithms: Based on backpropagation through time using stochastic gradient descent.
  - Shuffle the training corpus.
  - Break up the corpus into maxi-batches.
  - Break up each maxi-batch into mini-batches.
  - Compute gradients for each mini-batches.
  - Update all the parameters (weights and word embeddings) at the end of a maxi-batch.

Gradient descent optimization algorithms (Ruder 2016)

- SDG (stochastic gradient descent).
- SGD with momentum
- Adagrad (Adaptive Gradient)
- Adadelta (an extension of Adagrad)
- Adam (Adaptive Moment Estimation)
- NAG, RMSProp, AdaMax, Nadam, ...

- Implementation through computational graphs in TensorFlow, PyTorch, JAX.

Training issues

- Early stopping.
- Batch and layer normalization.
- Data augmentation.
- Hyperparameters
  - Dimension of source word embeddings.
  - Dimension of target word embeddings.
  - Number of LSTM/GRUs in encoder.
  - Number of LSTM/GRUs in decoder.
  - Number of layers in encoder of Transformer.
  - Number of layers in decoder of Transformer.
  - Initial learning rate.
  - Patience.
  - Dropout parameter.
  - Weight decay.
  - Noisy injection (to inputs, outputs, ...)
  - ...

Inference (decoding) with NMT (Koehn, 2017)

- For a source sentence x_1^J search for a target sentence $\hat{y}_1^{\hat{I}}$,

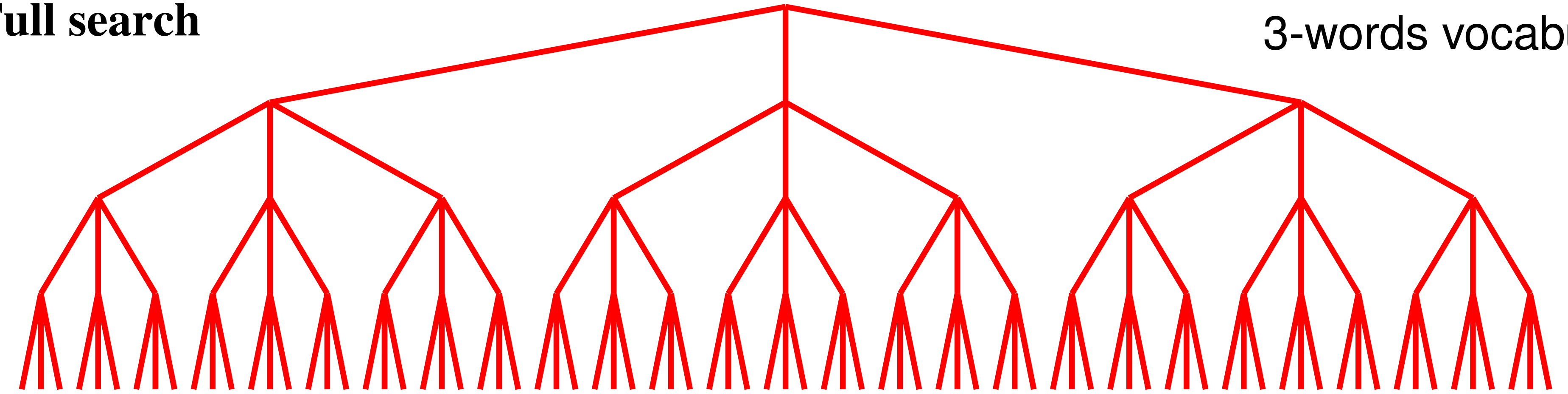
$$\begin{aligned}
\hat{y}_1^{\hat{I}} &= \operatorname*{argmax}_{I, y_1^I} p(y_1^I \mid x_1^J; \mathbf{W}) \\
&= \operatorname*{argmax}_{I, y_1^I} \prod_{i=1}^{I} p(y_i \mid y_1^{i-1}, u(x_1^J); \mathbf{W}) \\
&= \operatorname*{argmax}_{I, y_1^I} \prod_{i=1}^{I} \mathbf{f}_{sm}(\mathbf{W}_O\ \mathbf{h}_i^d(x_1^J))_{i(y_i)}
\end{aligned}$$

- Translation:
  - Greedy approach: for each i the maximum from the softmax output.
  - Smart tree search with beam search (suboptimal search)
    - At the begining of the process there is a empty list of target hypotheses.
    - At each target position there is a bounded prefix tree of K target prefixes. For each prefix, a bounded set of extended prefixes is generated by appending the most promising words.
    - The process ends when a “eof” is generated.

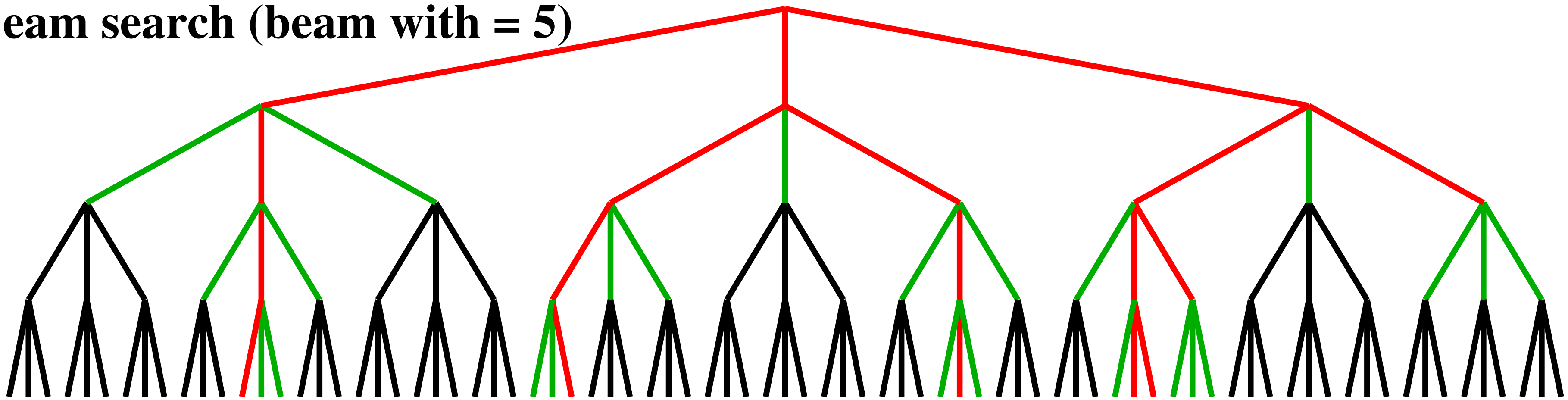
Decoding with NMT (Ney, 2018)

Full search

3-words vocabulary



Beam search (beam with = 5)



black=possible arcs **green = extended & discarded** **red = extended & preserved**

Index

1 Introduction ▷ 2

2 Syncronized feedforward neural networks ▷ 11

3 Syncronized recurrent neural networks ▷ 14

4 Neural machine translation ▷ 23

5 Attention models and recurrent neural networks ▷ 31

6 Attention models for all ▷ 41

7 Training and Decoding with NMT ▷ 67

∘ 8 *Translation results with neural machine translation* ▷ 73

9 Other issues of NMT ▷ 78

10 Bibliography ▷ 90

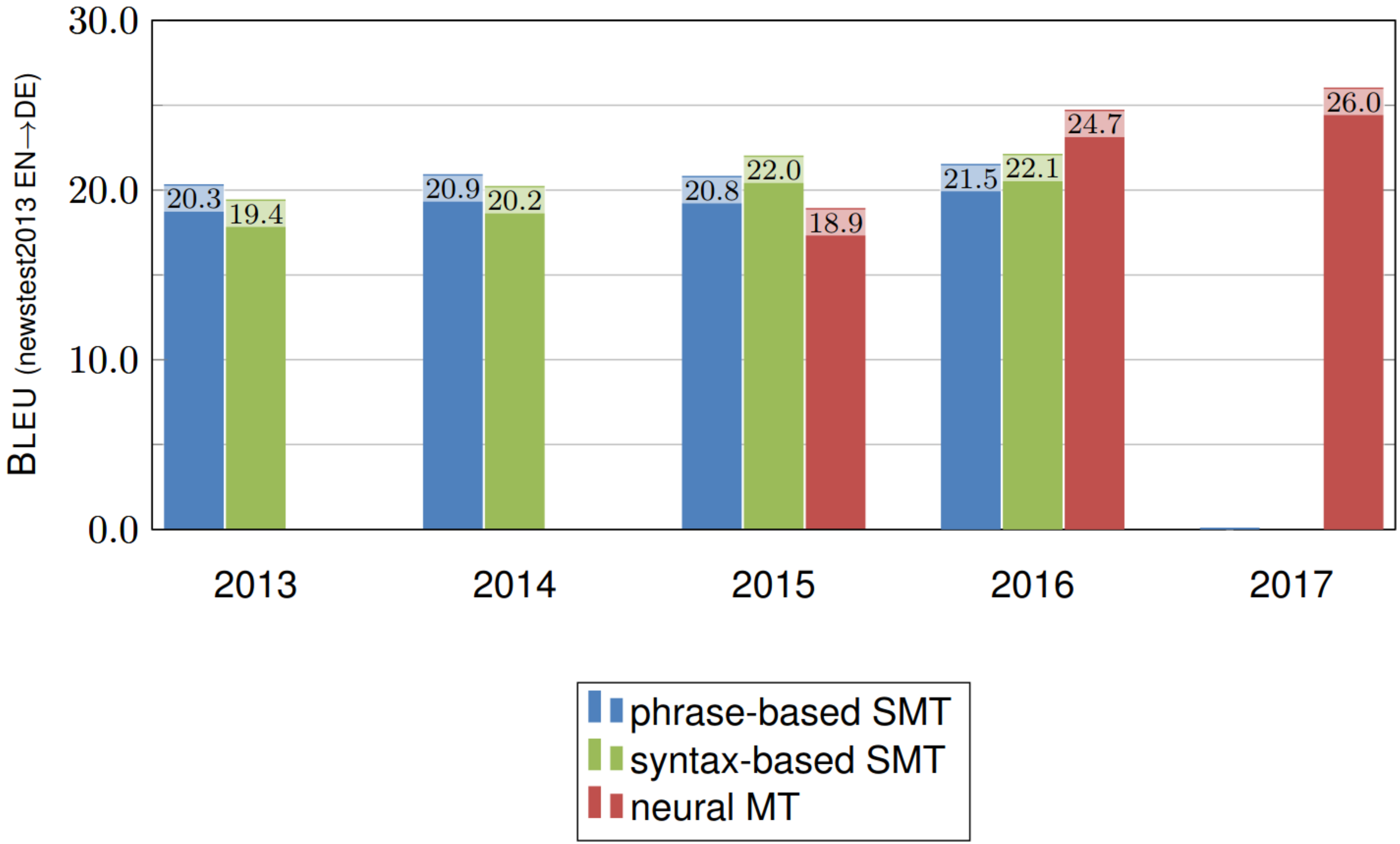
Corpus for text translation

- XRCE: Printer manuals.
- TED: Transcription of videos from expert speakers on education, business, science, tech and creativity.
  https://www.ted.com/talks
- UFAL: Medical texts.
  https://ufal.mff.cuni.cz/ufal_medical_corpus
- EUROPARL: Proceedings of the European Parliament
  https://www.statmt.org/europarl/

Text translation results (Peris 2019)

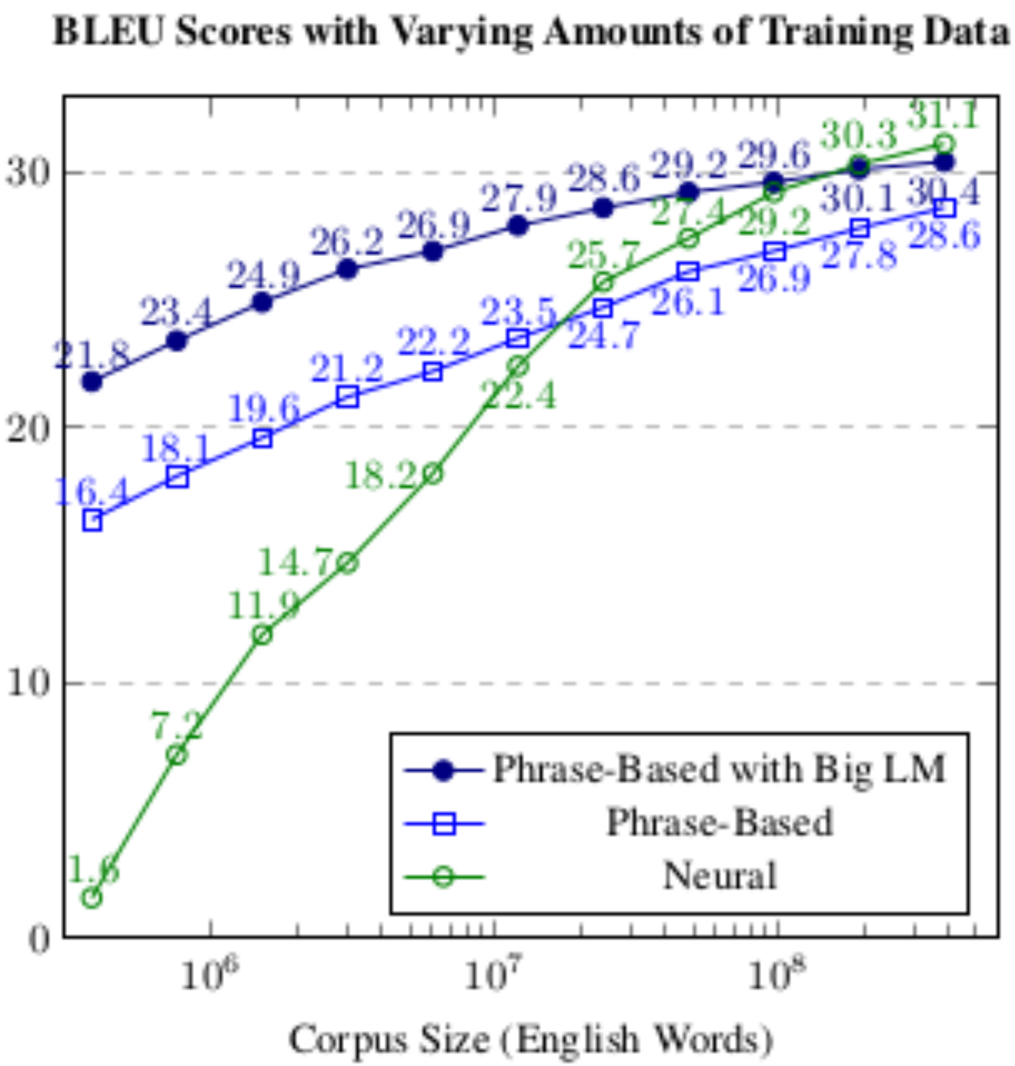
Task	Languages	Recurrent		Transformer		PB-SMT	
		BLEU	TER	BLEU	TER	BLEU	TER
XRCE	EN-FR	38.0	51.9	32.0	57.2	37.9	50.2
XRCE	GE-EN	36.2	51.1	31.3	54.9	36.8	50.1
TED	FR-EN	32.4	46.6	30.1	49.5	29.9	50.5
TED	ZH-EN	13.7	75.7	11.5	76.7	11.0	77.5
UFAL	EN-FR	37.2	46.1	37.8	45.9	35.0	47.9
UFAL	ES-EN	44.4	35.4	43.1	36.4	38.7	40.4
EUROPARL	EN-ES	25.0	58.6	26.1	55.4	24.6	56.8
EUROPARL	GE-EN	21.2	62.9	22.3	60.7	20.4	60.9

Edinburgh's WMT results over the years (Sennrich et al. 2017)



PBMT vs NMT and the size of training data (Koehn & Knowles. 2017)

English-Spanish



Index

1 Introduction ▷ 2

2 Syncronized feedforward neural networks ▷ 11

3 Syncronized recurrent neural networks ▷ 14

4 Neural machine translation ▷ 23

5 Attention models and recurrent neural networks ▷ 31

6 Attention models for all ▷ 41

7 Training and Decoding with NMT ▷ 67

8 Translation results with neural machine translation ▷ 73

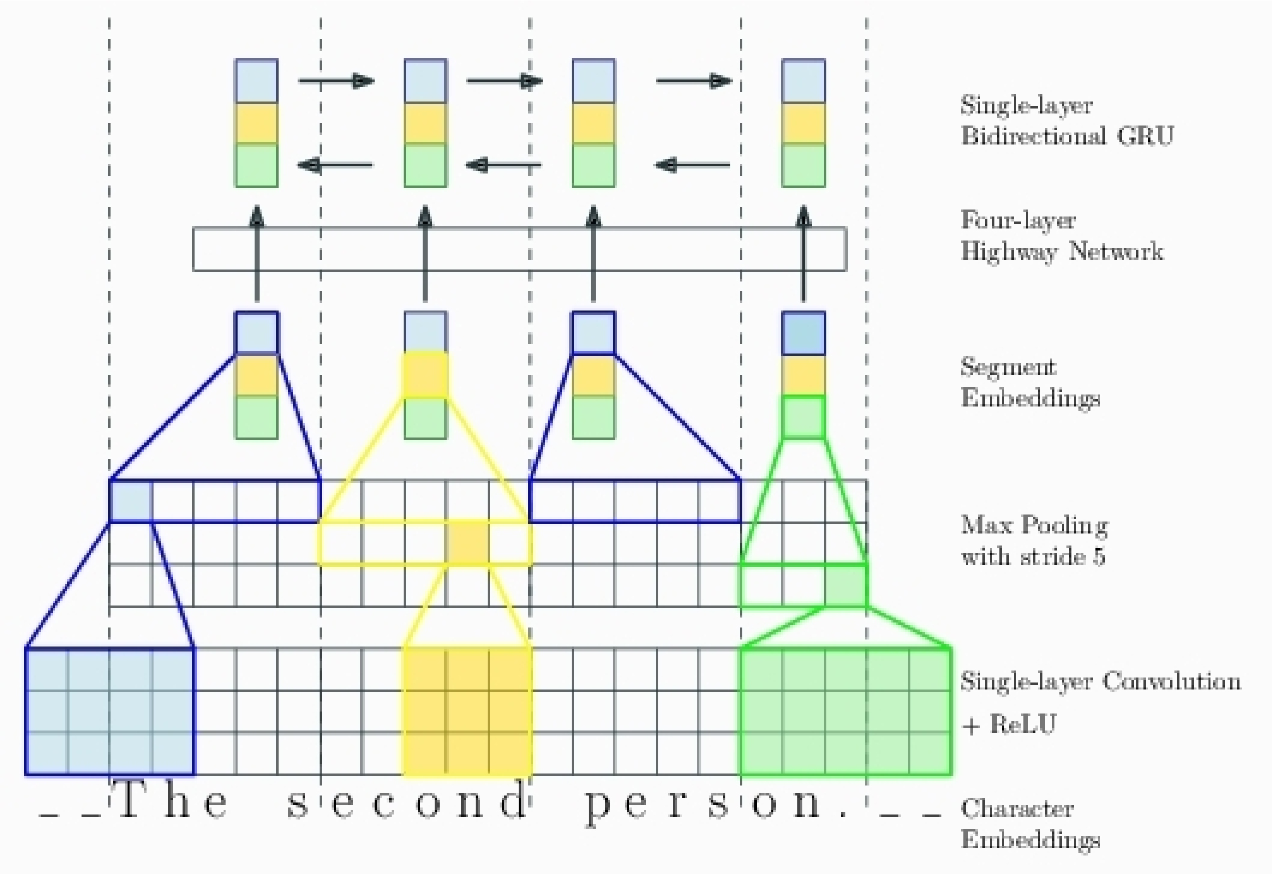
◦ 9 *Other issues of NMT* ▷ 78

10 Bibliography ▷ 90

Character based text translation (Larriba 2018)

- Pros:
  - Smaller vocabularies.
  - Reduce out-of-vocabulary words.
  - Profit from in-word lexical information
- Cons:
  - Longer input sequences.
  - References at word level.
- Approaches:
  - Fully character based NMT
  - Encoder based on convolutional character NN
  - Word embedding from character embedding

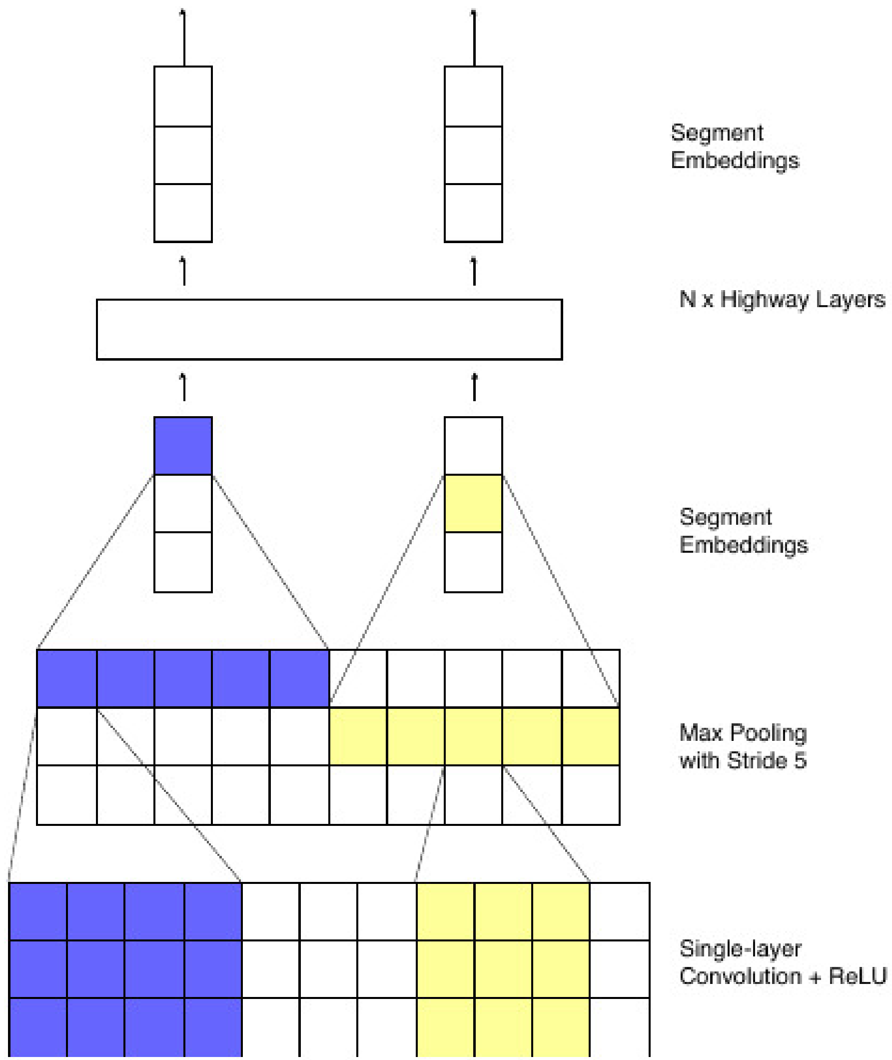
Convolutional character NMT (Lee 2017)



WMT'15 (Ge-En)

System	BLEU
BPE(s)+BPE(t) NN	24.0
BPE(s)+char(t) NN	25.3
char(s)+char(t) NN	25.8

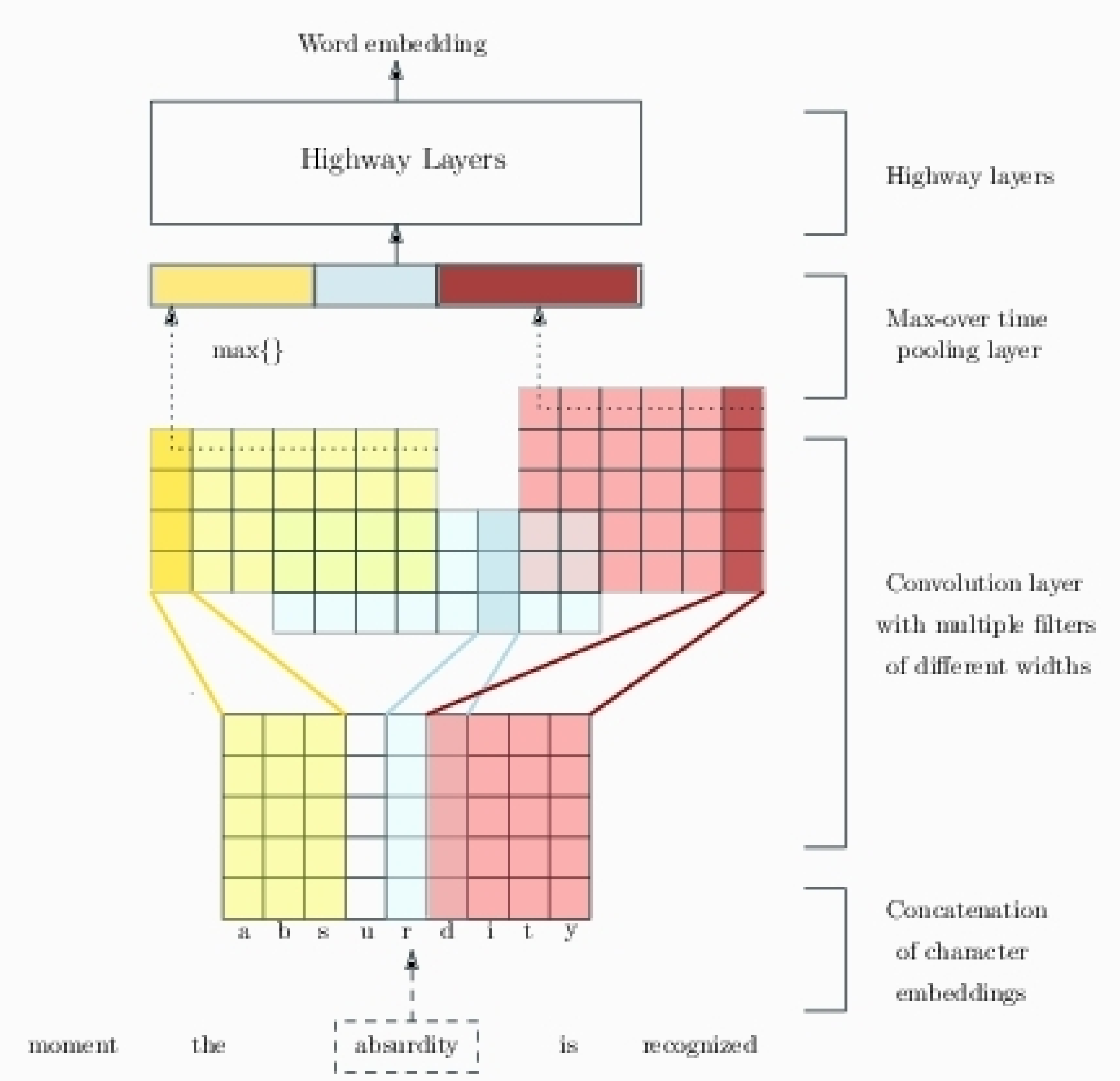
Convolutional character encoder for Transformer (Banar 2020)



WMT'15 (Ru-En)

System	Unit	BLEU
CharRNN	char	22.73
Transformer	char	26.99
CharTransformer	char	26.19
Transformer	bpe	28.01

Convolutional character encoder (Costa-Jussà & Fenollosa 2016)



	BLEU	
System	Ge-En	En-Ge
PBM	21.0	17.0
Word-based NN	20.6	17.2
Character-based NN	22.1	20.2

Refinements (Koehn, 2017)

- Ensemble decoding.
- Reranking n-best output list with a target language decoding.
- Large vocabularies and out-of-vocabulary words: BPE, categories, character-based translation, ...
- Adding monolingual data:
  - Back translation.
  - Adding a language model.
- Combining statistical alignment models and attention models.
- Adaptation.
- Adding linguistic annotation in the target.

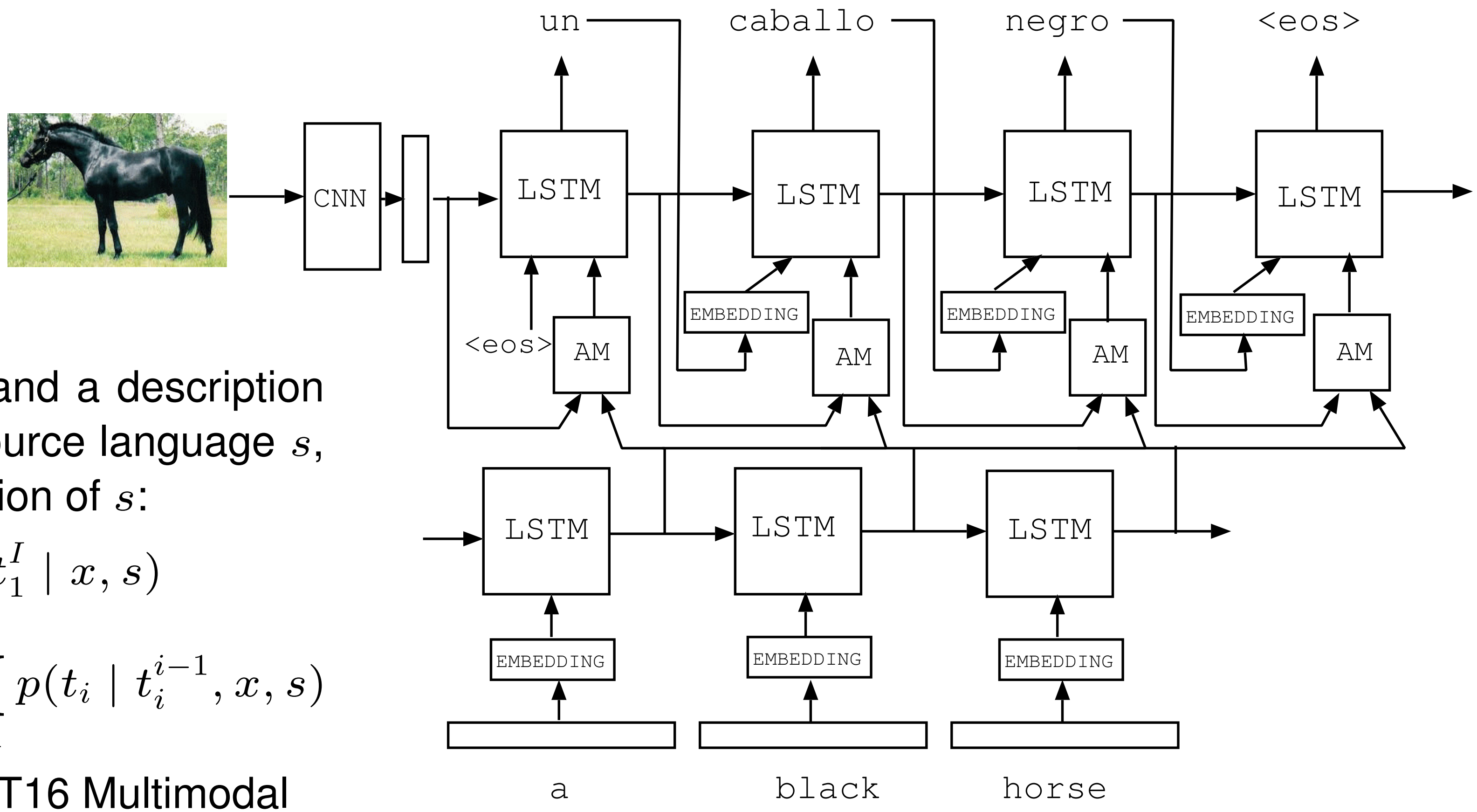
Other issues in the course

- Multimodal machine translation (in this section)
- Speech machine translation (in this section)
- Computer-assisted translation (topic 4)
- In topic 5:
  - Context-aware or document machine translation.
  - Multilingual machine translation.
  - Training NMT systems from monolingual corpora.
  - Syntax in NMT.
  - Reinforcement Learning in NMT.
  - Pre-trained models.

Other topics in the NMT

- Translation with low-resource languages.
- Data selection (this is a general topic)
- The use of specific glossaries (place holders)
- LegoNN: building encoder-decoder architectures with decoder modules that can be reused across various MT tasks [Damnia arXiv 2022]
- Reduction of inference time: CTranslate 2.0 from openNMT.
- Other computational issues: computational graphs, parallelism, the use of GPUs, ...

Multimodal machine translation (Huang 2016)



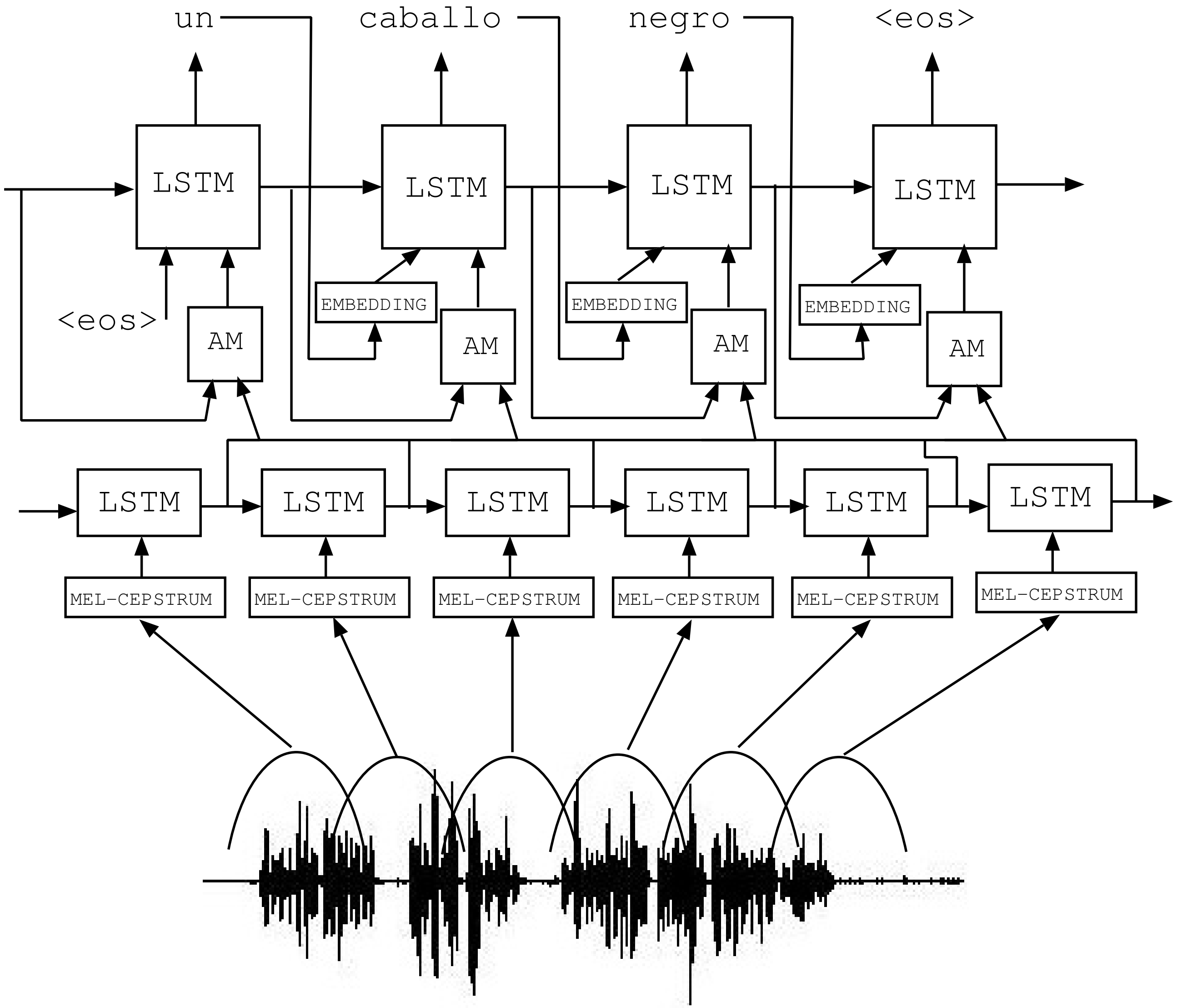
Given an image x and a description of the image in a source language s, search for a translation of s:

$$\hat{t}_1^{\hat{I}} = \operatorname*{argmax}_{I,t} p(t_1^I \mid x, s) = \operatorname*{argmax}_{I,t} \prod_{i=1}^{I} p(t_i \mid t_i^{i-1}, x, s)$$

Experiments: WMT16 Multimodal machine translation English-to-German

Systems	METEOR
Our model (without optimization)	41.1
Best system WMT16 (without Moses)	51.5
Best system WMT16 (using Moses)	53.2

Speech machine translation (Weiss et al. 2017)



Given an utterance x, search for a translation of s:

$$\hat{t}_1^{\hat{I}} = \underset{I,t}{\text{argmax}}\, p(t_1^I \mid x) = \underset{I,t}{\text{argmax}} \prod_{i=1}^{I} p(t_i \mid t_i^{i-1}, x)$$

Spanish-English Fisher task

Systems	BLEU
End-to-end	47.3
Supervised source	48.7
Cascade ASR+NMT	45.5

Toolkits for neural machine translation

- Fairseq https://github.com/pytorch/fairseq
  Facebook, Google Brain (Ott+ arXiv 2019)
- MarianNMT https://github.com/marian-nmt/marian/
  Microsoft, Adam Mickiewicz U., U. of Edinburgh, (Junczys-Dowmunt+ ACL 2018)
- modernNMT https://github.com/modernmt/modernmt
  Facebook, FBK. Translated ()
- Nematus https://github.com/EdinburghNLP/nematus
  U. of Edinburgh, Middle East U., New York U., Heidelberg U., U. of Zurich (Sennrich+ arXiv 2017)
- NMT-Keras https://github.com/lvapeab/nmt-keras
  UPV (Peris+ PBML 2018)
- OpenNMT https://github.com/opennmt
  SYSTRAN, Harward U., Ubiqus (Klein+ arXiv 2018)
- RETURNN https://github.com/rwth-i6/returnn
  RWTH Aachen, AppTek, NNAISENSE (Zeyer+ arXiv 2018)
- Sockeye https://github.com/awslabs/sockeye
  Amazon (Hieber+ arXiv 2017)
- T2T https://github.com/tensorflow/tensor2tensor
  Google, DeepMind (Wasvani+ arXiv 2018)
- XNMT https://github.com/neulab/xnmt
  CMU, KIT, Limsi, NIST, Inria, Pennsylvania U., Illinois U. (Neubig+ arXiv 2018)

Neural machine translation systems

- DeepL https://www.deepl.com/en/translator
- Google translate https://translate.google.com/
- Pure Neural Machine Translation https://translate.systran.net/translationTools/text
- UPV Neural Machine Translation http://casmacat.prhlt.upv.es/inmt/

Index

1 Introduction ▷ 2

2 Syncronized feedforward neural networks ▷ 11

3 Syncronized recurrent neural networks ▷ 14

4 Neural machine translation ▷ 23

5 Attention models and recurrent neural networks ▷ 31

6 Attention models for all ▷ 41

7 Training and Decoding with NMT ▷ 67

8 Translation results with neural machine translation ▷ 73

9 Other issues of NMT ▷ 78

◦ 10 *Bibliography* ▷ 90

Bibliography (1)

- Banar et al. Character-level Transformer-based Neural Machine Translation. NLPIR 2020.
- Bahdanau et al. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473. 2015
- Bengio et al. A Neural Probabilistic Language Model, JMLR 2003.
- Castaño & Casacuberta. Preliminary Experimentations for ASR Understanding Through Simple RN. Eurospeech 1995
- Castaño & Casacuberta. A Connectionist Approach to Machine translation. Eurospeech 1997.
- Castaño & Casacuberta. Text-to-text machine translation using the RECONTRA connectionist model. IWANN 1999.
- Chollampatt, Hendy Susanto, Tan, Szymanska. Can Automatic Post-Editing Improve NMT?. EMNLP 2020.
- Chung, Gulcehre, Cho, Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv:1412.3555. 2014
- Costa-Jusa & Fenollosa. Character-based neural machine translation. ACL. 2016.
- Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019.
- Elliot, Frank & Hasler. Multilingual Image Description with Neural Sequence Models. ICLR. 2016.

Bibliography (2)

- Gers & Schmidhuber. Recurrent nets that time and count. International Joint Conference on Neural Networks. 2000.
- Hieber et al. Sockeye: A Toolkit for Neural Machine Translation.arXiv. 2017.
- Hochreiter & Schmidhuber. Long short-term memory. Neural Computation. 1997.
- Huang et al. Attention-based Multimodal Neural Machine Translation. CMT 2016.
- Koehn. Neural Machine Translation. arXiv:1709.07809v1. 2017.
- Koehn. Neural Machine Translation. Cambridge Univ. 2020.
- Junczys-Dowmunt et al. Marian: Fast Neural Machine Translation in C++. ACL 2018.
- Klein et al. OpenNMT: Open-Source Toolkit for Neural Machine Translation. ACL 2017.
- Koehn & Knowles. Six Challenges for Neural Machine Translation. WNMT. 2017.
- Kratzer, Toussaint, Mainprice. Prediction of Human Full-Body Movements with Motion Optimization and Recurrent Neural Networks. IEEE ICRA. 2020.
- Lakew et al. A Comparison of Transformer and Recurrent Neural Networks on Multilingual Neural Machine Translation. COLING 2018.
- Larriba. Character-based Neural Machine Translation. Seminar PRHLT. 2018.
- Lee et al. Fully Character-Level Neural Machine Translation without Explicit Segmentation. TACL. 2017.

Bibliography (3)

- Luong et al. Effective Approaches to Attention-based Neural Machine Translation. arXiv:1508.04025. 2015.
- Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality. arXiv:1310.4546. 2013
- Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality. NIPS 2013.
- Neubig et al. XNMT: The eXtensible Neural Machine Translation Toolkit. AMTA 2018.
- Ney. Speech Recognition and Machine Translation: From Statistical Decison Theory to Machine Learning and Deep Neural Networks. 2nd International Summer School on Deep Learning. 2018
- Ott et al. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. NAACL 2019.
- Oeris & Casacuberta. NMT-Keras: a Very Flexible Toolkit with a Focus on Interactive NMT and Online Learning. PBML 2018.
- Peris. Interactivity, adaptation and multimodality in neural sequence-to-sequence learning. Ph.D. Thesis. 2019.
- Peris et al. Video Description Using Bidirectional Recurrent Neural Networks. ICANN. 2016
- Peris & Casacuberta. Interactive neural machine translation. CSL. 2017.
- Ruder. An overview of gradient descent optimization algorithms. arXiv, 2016

Bibliography (4)

- Schuster & Paliwal. Bidirectional Recurrent Neural Networks. IEEE Transactions on Signal Processing. 1997.
- Sennrich et al. Neural Machine Translation of Rare Words with Subword Units. arXiv:1508.07909. 2015.
- Sennrich et al. Advances in Neural Machine Translation. AMTA. 2016.
- Sundermeyer. Improvements in Language and Translation Modeling, PhD. 2015
- Sutskever et al. Sequence to Sequence Learning with Neural Networks. arXiv 2014.
- Vinyals et al. Show and Tell: A Neural Image Caption Generator. CVPR. 2015
- Vaswani et al. Attention Is All You Need. arXiv 2017.
- Weiss et al. Sequence-to-Sequence Models Can Directly Translate Foreign Speech. arXiv 2017.
- Xiong et al. On Layer Normalization in the Transformer Architecture. arXiv 2020.
- Yao et al. Describing Videos by Exploiting Temporal Structure. arXiv:1502.08029.2015.
- Zeyer et al. RETURNN as a Generic Flexible Neural Toolkit with Application to Translation and Speech Recognition. ACL. 2018.