THE UNIVERSITY of EDINBURGH

Practical Neural Machine Translation

Rico Sennrich, Barry Haddow

Institute for Language, Cognition and Computation
University of Edinburgh

April 4, 2017
(Last Updated: April 28, 2017)

Practical Neural Machine Translation

NMT Timeline

1987 Early encoder-decoder, with vocabulary size 30-40 [Allen, 1987]

⋮

2013 Pure neural MT system presented [Kalchbrenner and Blunsom, 2013]

2014 Competitive encoder-decoder for large-scale MT
[Bahdanau et al., 2015, Luong et al., 2014]

2015 NMT systems in shared tasks – perform well in WMT, state-of-the-art at IWSLT

2016 NMT systems top most language pairs in WMT

2016 Commercial deployments of NMT launched

NMT now state-of-the-art

system	BLEU	official rank
uedin-nmt	34.2	1
metamind	32.3	2
uedin-syntax	30.6	3
NYU-UMontreal	30.8	4
online-B	29.4	5-10
KIT/LIMSI	29.1	5-10
cambridge	30.6	5-10
online-A	29.9	5-10
promt-rule	23.4	5-10
KIT	29.0	6-10
jhu-syntax	26.6	11-12
jhu-pbmt	28.3	11-12
uedin-pbmt	28.4	13-14
online-F	19.3	13-15
online-G	23.8	14-15

WMT16 EN→DE

system	BLEU	official rank
uedin-nmt	38.6	1
online-B	35.0	2-5
online-A	32.8	2-5
uedin-syntax	34.4	2-5
KIT	33.9	2-6
uedin-pbmt	35.1	5-7
jhu-pbmt	34.5	6-7
online-G	30.1	8
jhu-syntax	31.0	9
online-F	20.2	10

WMT16 DE→EN

NMT now state-of-the-art

system	BLEU	official rank
uedin-nmt	34.2	1
metamind	32.3	2
uedin-syntax	30.6	3
NYU-UMontreal	30.8	4
online-B	29.4	5-10
KIT/LIMSI	29.1	5-10
cambridge	30.6	5-10
online-A	29.9	5-10
promt-rule	23.4	5-10
KIT	29.0	6-10
jhu-syntax	26.6	11-12
jhu-pbmt	28.3	11-12
uedin-pbmt	28.4	13-14
online-F	19.3	13-15
online-G	23.8	14-15

WMT16 EN→DE

system	BLEU	official rank
uedin-nmt	38.6	1
online-B	35.0	2-5
online-A	32.8	2-5
uedin-syntax	34.4	2-5
KIT	33.9	2-6
uedin-pbmt	35.1	5-7
jhu-pbmt	34.5	6-7
online-G	30.1	8
jhu-syntax	31.0	9
online-F	20.2	10

WMT16 DE→EN

- pure NMT

NMT now state-of-the-art

system	BLEU	official rank
uedin-nmt	34.2	1
metamind	32.3	2
uedin-syntax	30.6	3
NYU-UMontreal	30.8	4
online-B	29.4	5-10
KIT/LIMSI	29.1	5-10
cambridge	30.6	5-10
online-A	29.9	5-10
promt-rule	23.4	5-10
KIT	29.0	6-10
jhu-syntax	26.6	11-12
jhu-pbmt	28.3	11-12
uedin-pbmt	28.4	13-14
online-F	19.3	13-15
online-G	23.8	14-15

WMT16 EN→DE

system	BLEU	official rank
uedin-nmt	38.6	1
online-B	35.0	2-5
online-A	32.8	2-5
uedin-syntax	34.4	2-5
KIT	33.9	2-6
uedin-pbmt	35.1	5-7
jhu-pbmt	34.5	6-7
online-G	30.1	8
jhu-syntax	31.0	9
online-F	20.2	10

WMT16 DE→EN

- pure NMT
- NMT component

NMT now state-of-the-art

uedin-nmt	25.8	1
NYU-UMontreal	23.6	2
jhu-pbmt	23.6	3
cu-chimera	21.0	4-5
cu-tamchyna	20.8	4-5
uedin-cu-syntax	20.9	6-7
online-B	22.7	6-7
online-A	19.5	15
cu-TectoMT	14.7	16
cu-mergedtrees	8.2	18

WMT16 EN→CS

uedin-nmt	31.4	1
jhu-pbmt	30.4	2
online-B	28.6	3
PJATK	28.3	8-10
online-A	25.7	11
cu-mergedtrees	13.3	12

WMT16 CS→EN

online-B	39.2	1-2
uedin-nmt	33.9	1-2
uedin-pbmt	35.2	3
uedin-syntax	33.6	4-5
online-A	30.8	4-6
jhu-pbmt	32.2	5-7
LIMSI	31.0	6-7

WMT16 RO→EN

uedin-nmt	28.1	1-2
QT21-HimL-SysComb	28.9	1-2
KIT	25.8	3-7
uedin-pbmt	26.8	3-7
online-B	25.4	3-7
uedin-lmu-hiero	25.9	3-7
RWTH-SYSCOMB	27.1	3-7
LIMSI	23.9	8-10
lmu-cuni	24.3	8-10
jhu-pbmt	23.5	8-11
usfd-rescoring	23.1	10-12
online-A	19.2	11-12

WMT16 EN→RO

NMT now state-of-the-art

PROMT-rule	22.3	1
amu-uedin	25.3	2-4
online-B	23.8	2-5
uedin-nmt	26.0	2-5
online-G	26.2	3-5
NYU-UMontreal	23.1	6
jhu-pbmt	24.0	7-8
LIMSI	23.6	7-10
online-A	20.2	8-10
AFRL-MITLL-phr	23.5	9-10
AFRL-MITLL-verb	20.9	11
online-F	8.6	12

WMT16 EN→RU

uedin-pbmt	23.4	1-4
online-G	20.6	1-4
online-B	23.6	1-4
UH-opus	23.1	1-4
PROMT-SMT	20.3	5
UH-factored	19.3	6-7
uedin-syntax	20.4	6-7
online-A	19.0	8
jhu-pbmt	19.1	9

WMT16 FI→EN

amu-uedin	29.1	1-2
online-G	28.7	1-3
NRC	29.1	2-4
online-B	28.1	3-5
uedin-nmt	28.0	4-5
online-A	25.7	6-7
AFRL-MITLL-phr	27.6	6-7
AFRL-MITLL-contrast	27.0	8-9
PROMT-rule	20.4	8-9
online-F	13.5	10

WMT16 RU→EN

online-G	15.4	1-3
abumatra-nmt	17.2	1-4
online-B	14.4	1-4
abumatran-combo	17.4	3-5
UH-opus	16.3	4-5
NYU-UMontreal	15.1	6-8
abumatran-pbsmt	14.6	6-8
online-A	13.0	6-8
jhu-pbmt	13.8	9-10
UH-factored	12.8	9-12
aalto	11.6	10-13
jhu-hltcoe	11.9	10-13
UUT	11.6	11-13

WMT16 EN→FI

NMT in Production

SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market

NMT in Production

SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market

Google announces Neural Machine Translation to improve Google Translate

NMT in Production

SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market

Google announces Neural Machine Translation to improve Google Translate

WIPO goes Neural

Oct 4, 2016 | 590 views | 41 Likes | 3 Comments

NMT in Production

SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market

Google announces Neural Machine Translation to improve Google Translate

WIPO goes Neural

Oct 4, 2016 | 590 views | 41 Likes | 3 Comments

Microsoft Translator launching Neural Network based translations for all its speech languages

Rate this article ★★★★★

Microsoft Translator November 15, 2016

Share 462 | 285 | 0 | 0

NMT in Production

SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market

Baidu Says They Were First in NMT: Language Industry News Roundup

by Marion Marking on January 16, 2017

Andrew Ng, Stanford Adjunct Professor, Coursera co-Founder, and Chief Scientist at Chinese search giant Baidu, recently made a bold claim that may raise a few eyebrows in the NMT community.

Google announces Neural Machine Translation to improve Google Translate

WIPO goes Neural

Oct 4, 2016 | 590 views | 41 Likes | 3 Comments

Microsoft Translator launching Neural Network based translations for all its speech languages

Rate this article ★★★★★

Microsoft Translator November 15, 2016

Share 462 | 285 | 0 | 0

NMT in Production

SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market

Baidu Says They Were First in NMT: Language Industry News Roundup

by Marion Marking on January 16, 2017

Andrew Ng, Stanford Adjunct Professor, Coursera co-Founder, and Chief Scientist at Chinese search giant Baidu, recently made a bold claim that may raise a few eyebrows in the NMT community.

Google announces Neural Machine Translation to improve Google Translate

WIPO goes Neural

Oct 4, 2016 | 590 views | 41 Likes | 3 Comments

Microsoft Translator launching Neural Network based translations for all its speech languages

Rate this article ★★★★★

Microsoft Translator November 15, 2016

Share 462 | 285 | 0 | 0

TraMOOC
Translation for Massive Open Online Courses

NMT in Production

SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market

Baidu Says They Were First in NMT: Language Industry News Roundup

by Marion Marking on January 16, 2017

Andrew Ng, Stanford Adjunct Professor, Coursera co-Founder, and Chief Scientist at Chinese search giant Baidu, recently made a bold claim that may raise a few eyebrows in the NMT community.

Google announces Neural Machine Translation to improve Google Translate

WIPO goes Neural

Oct 4, 2016 | 590 views | 41 Likes | 3 Comments

Microsoft Translator launching Neural Network based translations for all its speech languages

Rate this article ★★★★★

Microsoft Translator November 15, 2016

Share 462 | 285 | 0 | 0

TraMOOC
Translation for Massive Open Online Courses

Health in my Language
HimL

NMT in Production

SYSTRAN announces the launch of its "Purely Neural MT" engine, a revolution for the machine translation market

Baidu Says They Were First in NMT: Language Industry News Roundup

by Marion Marking on January 16, 2017

Andrew Ng, Stanford Adjunct Professor, Coursera co-Founder, and Chief Scientist at Chinese search giant Baidu, recently made a bold claim that may raise a few eyebrows in the NMT community.

Google announces Neural Machine Translation to improve Google Translate

WIPO goes Neural

Oct 4, 2016 | 590 views | 41 Likes | 3 Comments

Microsoft Translator launching Neural Network based translations for all its speech languages

Rate this article ★★★★★

Microsoft Translator November 15, 2016

Share 462 | 285 | 0 | 0

TraMOOC
Translation for Massive Open Online Courses

Health in my Language

HimL

SUMMA

Course Goals

At the end of this tutorial, you will

- have a basic theoretical understanding of models/algorithms in NMT
- understand strengths and weaknesses of NMT
- know techniques that help to build state-of-the-art NMT systems
- know practical tips for various problems you may encounter:
  - training and decoding efficiency
  - domain adaptation
  - ways to further improve translation quality
  - ...

no hands-on coding/training in tutorial, but helpful resources are provided

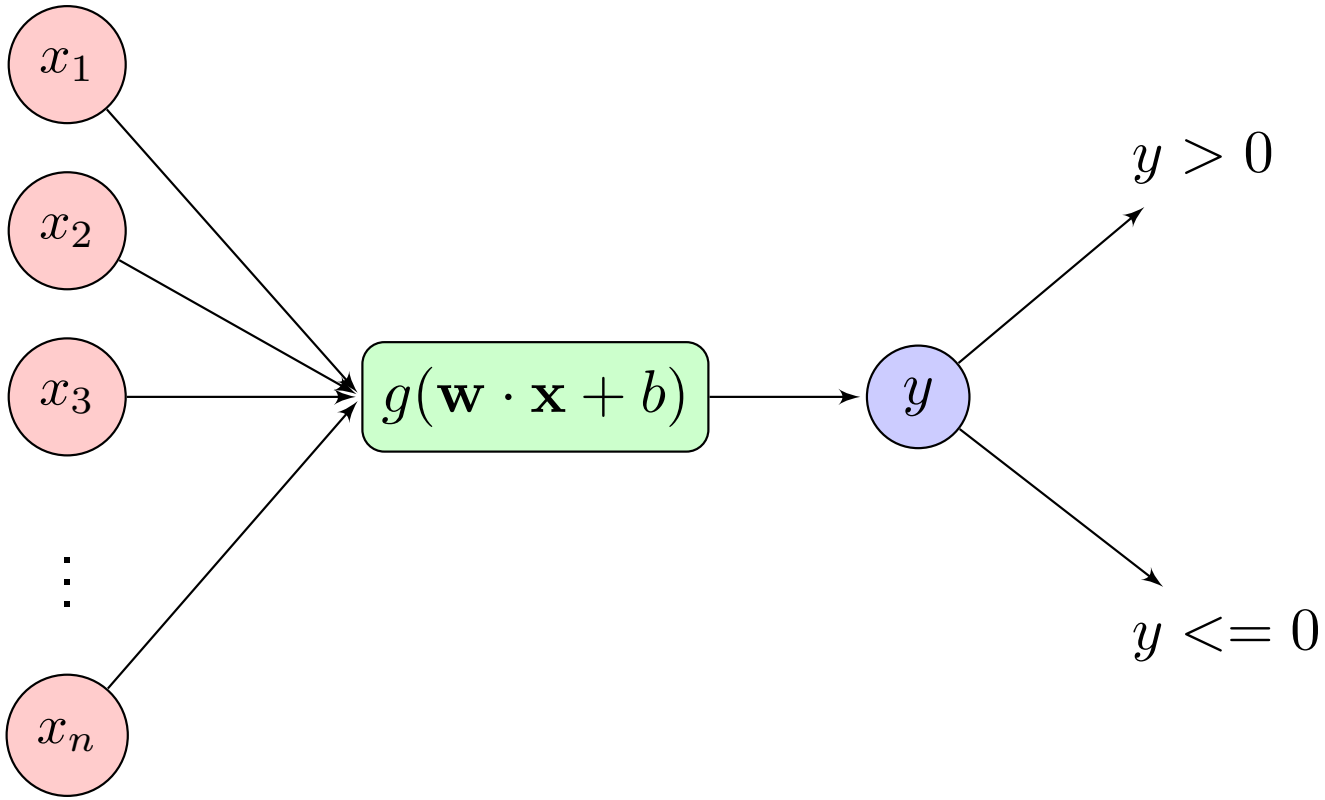
What is a Neural Network?

- A complex non-linear function which:
  - is built from simpler units (neurons, nodes, gates, . . .)
  - maps vectors/matrices to vectors/matrices
  - is parameterised by vectors/matrices

What is a Neural Network?

- A complex non-linear function which:
  - is built from simpler units (neurons, nodes, gates, . . .)
  - maps vectors/matrices to vectors/matrices
  - is parameterised by vectors/matrices
- Why is this useful?
  - very expressive
  - can represent (e.g.) parameterised probability distributions
  - evaluation and parameter estimation can be built up from components

A Simple Neural Network Classifier



- $\mathbf{x}$ is a vector input, y is a scalar output
- $\mathbf{w}$ and b are the parameters (b is a bias term)
- g is a non-linear activation function

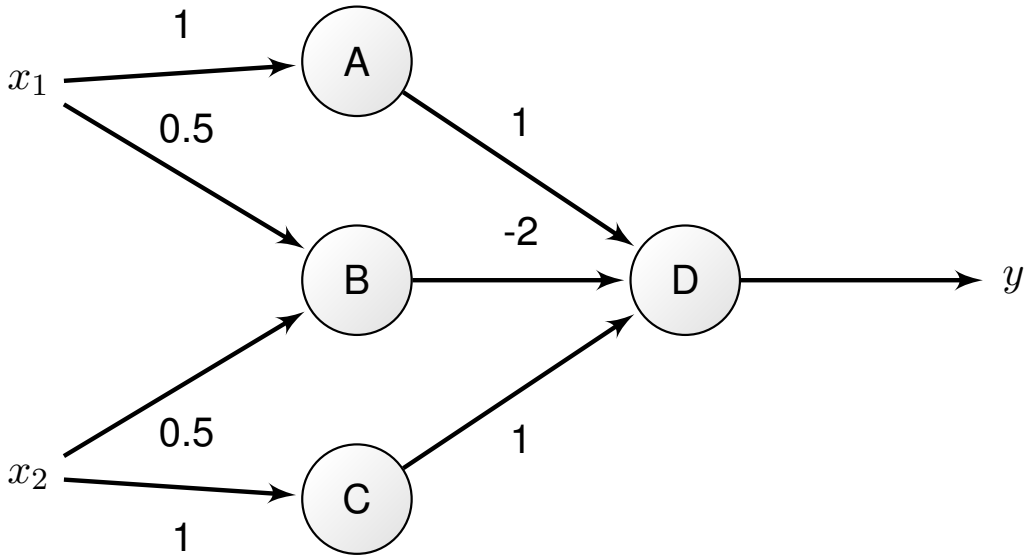
Why Non-linearity?

Functions like XOR cannot be separated by a linear function

XOR

Truth table

x_1	x_2	output
0	0	0
0	1	1
1	0	1
1	1	0



(neurons arranged in layers, and fire if input is ≥ 1)

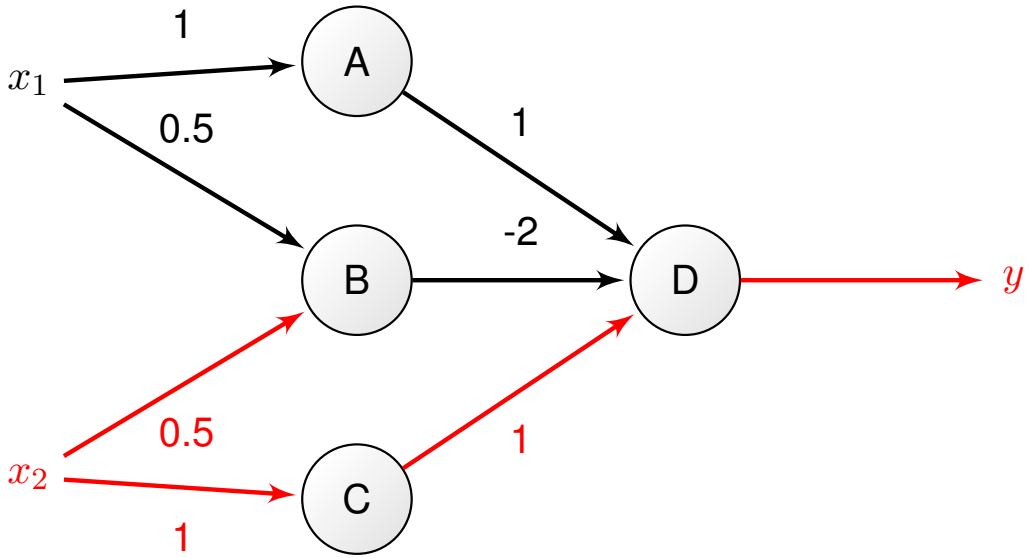
Why Non-linearity?

Functions like XOR cannot be separated by a linear function

XOR

Truth table

x_1	x_2	output
0	0	0
0	1	1
1	0	1
1	1	0



(neurons arranged in layers, and fire if input is ≥ 1)

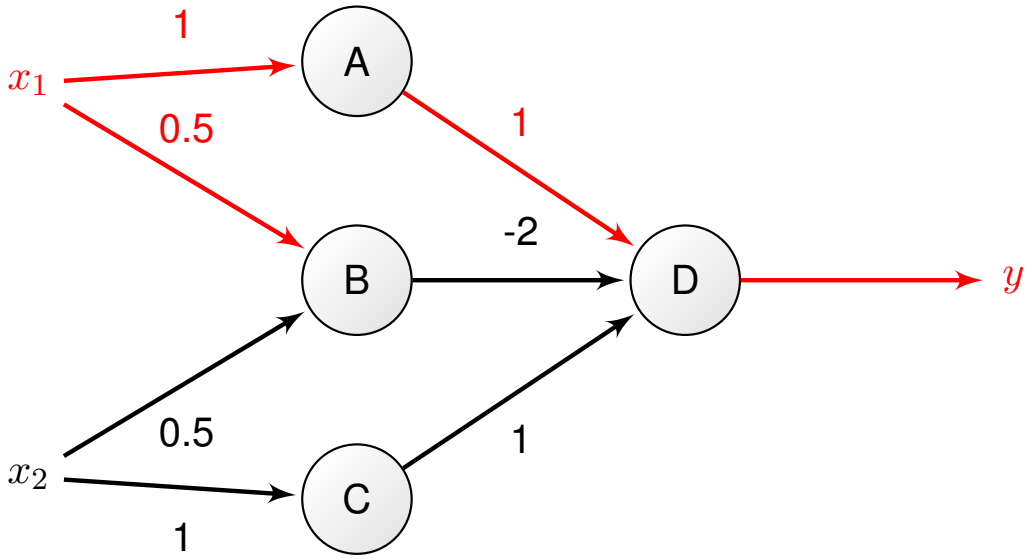
Why Non-linearity?

Functions like XOR cannot be separated by a linear function

XOR

Truth table

x_1	x_2	output
0	0	0
0	1	1
1	0	1
1	1	0



(neurons arranged in layers, and fire if input is ≥ 1)

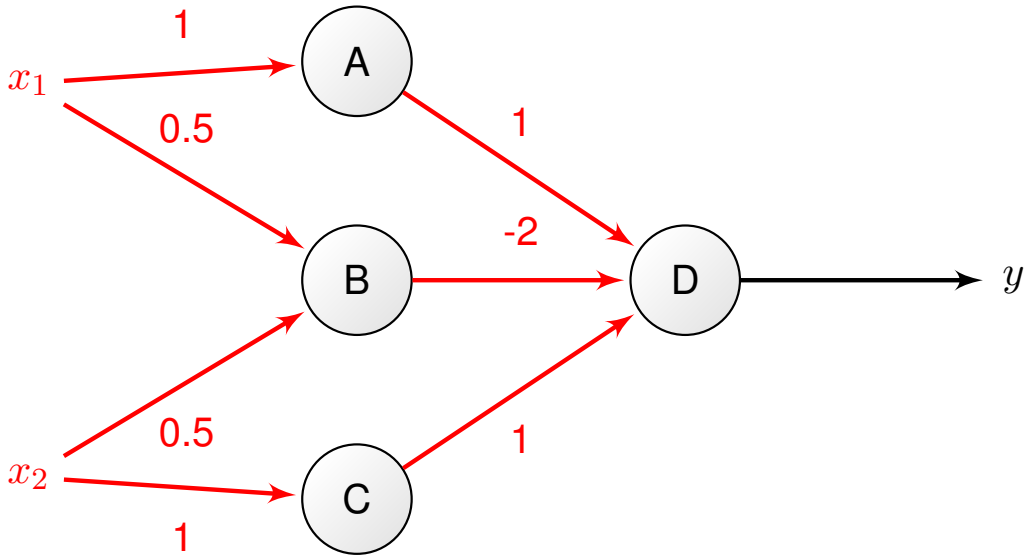
Why Non-linearity?

Functions like XOR cannot be separated by a linear function

XOR

Truth table

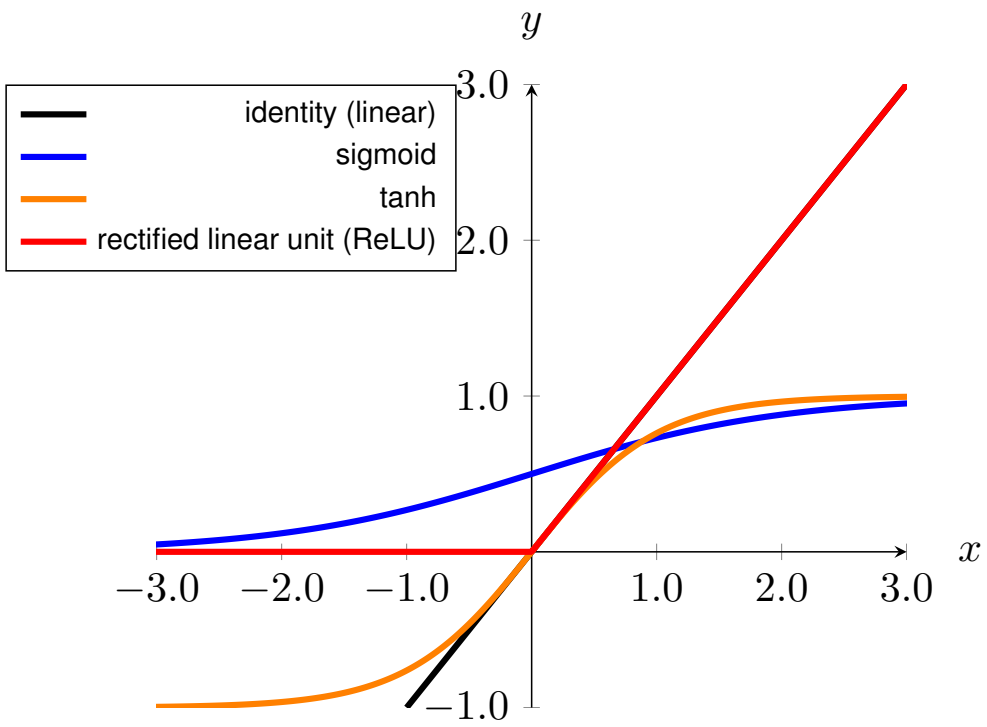
x_1	x_2	output
0	0	0
0	1	1
1	0	1
1	1	0



(neurons arranged in layers, and fire if input is ≥ 1)

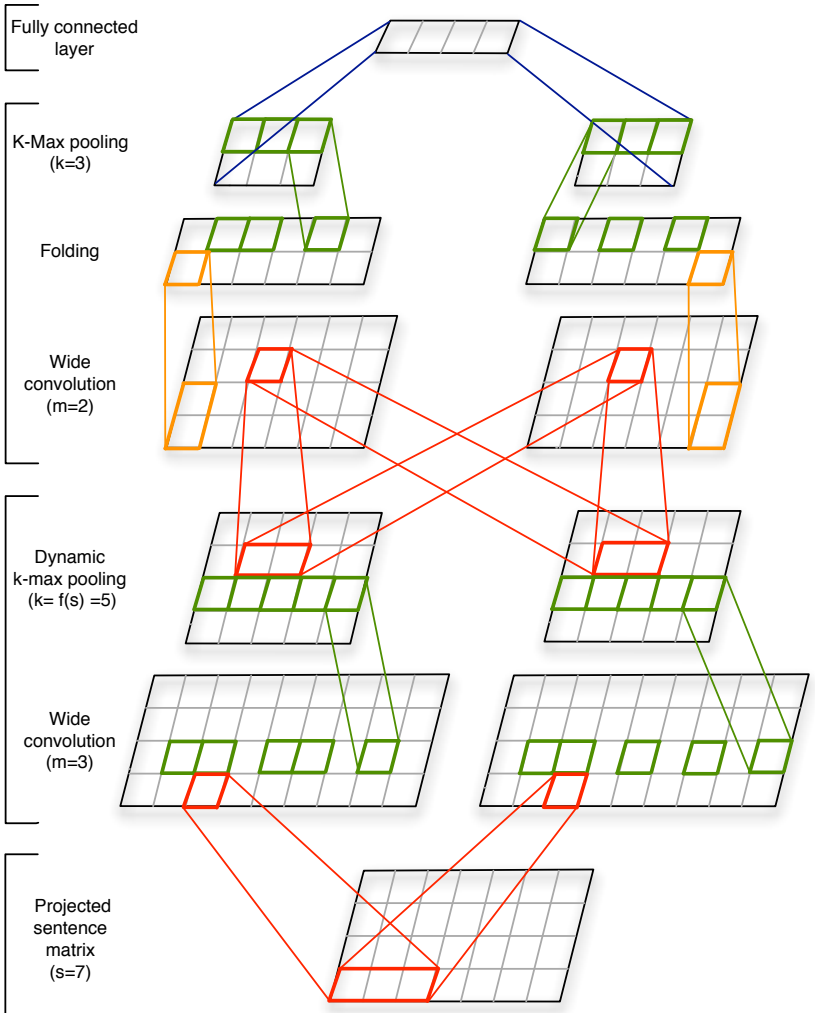
Activation functions

- desirable:
  - differentiable (for gradient-based training)
  - monotonic (for better training stability)
  - non-linear (for better expressivity)



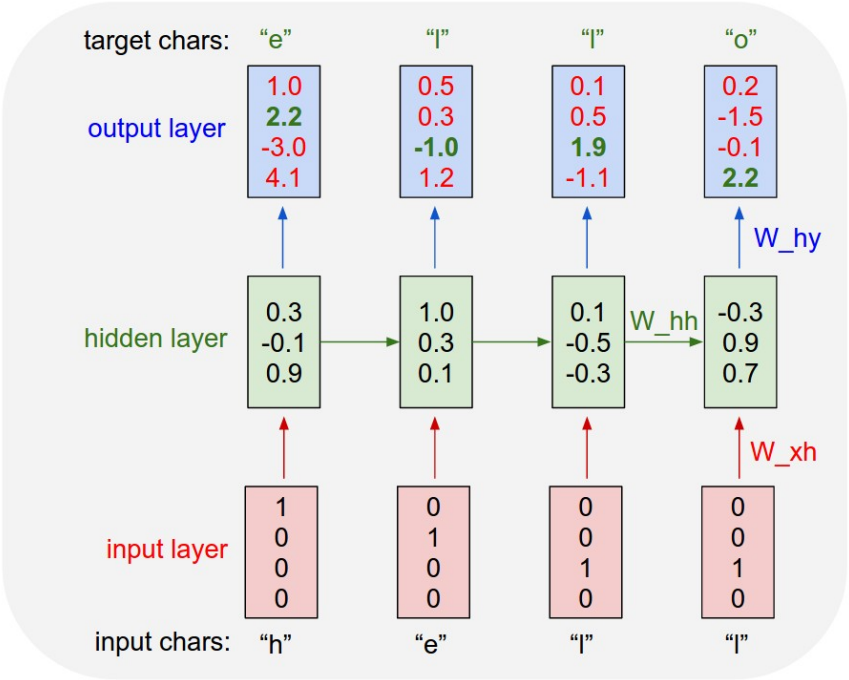
More Complex Architectures

Convolutional



[Kalchbrenner et al., 2014]

Recurrent



Andrej Karpathy

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

Training of Neural Networks

- Parameter estimation
  - Use gradient descent
  - Requires labelled training data . . .
    . . . and differentiable objective function
- Network structure enables efficient computation
  - Forward pass to compute network output
  - Backpropogation, i.e. backward pass using chain rule, to calculate gradient
- Normally train stochastically using mini-batches

Practical Considerations

- hyperparameters:
  - number and size of layers
  - minibatch size
  - learning rate
  - ...
- initialisation of weight matrices
- stopping criterion
- regularization (dropout)
- bias units (always-on input)

Toolkits for Neural Networks

What does a Toolkit Provide

- Multi-dimensional matrices (tensors)
- Automatic differentiation
- Efficient GPU routines for tensor operations

`http://torch.ch/`

TensorFlow `https://www.tensorflow.org/`

`http://deeplearning.net/software/theano/`

There are many more!

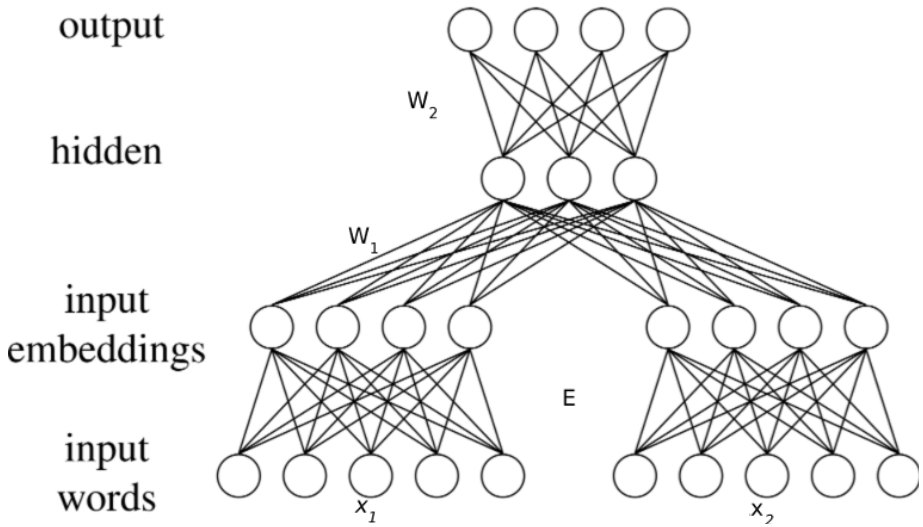
Language model

chain rule and Markov assumption

- a sentence T of length n is a sequence $w_1, \ldots, w_n$

$$
\begin{aligned}
p(T) &= p(w_1, \ldots, w_n) \\
&= \prod_{i=1}^{n} p(w_i | w_0, \ldots, w_{i-1}) && \text{(chain rule)} \\
&\approx \prod_{i=1}^{n} p(w_i | w_{i-k}, \ldots, w_{i-1}) && \text{(Markov assumption: n-gram model)}
\end{aligned}
$$

N-gram language model with feedforward neural network



[Vaswani et al., 2013]

n-gram NNLM [Bengio et al., 2003]

- input: context of n-1 previous words
- output: probability distribution for next word
- linear **embedding layer** with shared weights
- one or several **hidden layers**

Representing words as vectors

One-hot encoding

- example vocabulary: 'man, 'runs', 'the', '.'
- input/output for $p(\text{runs}|\text{the man})$:

$$x_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad x_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad y_{\text{true}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

- size of input/output vector: vocabulary size
- embedding layer is lower-dimensional and dense
  - smaller weight matrices
  - network learns to group similar words to similar point in vector space

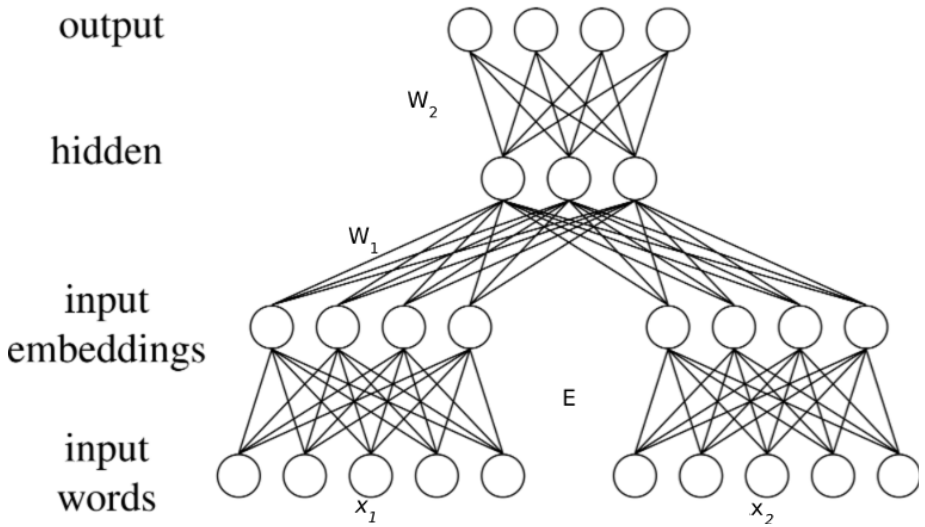
Softmax activation function

softmax function

$$p(y = j|x) = \frac{e^{x_j}}{\sum_k e^{x_k}}$$

- softmax function normalizes output vector to probability distribution
  $\rightarrow$ computational cost linear to vocabulary size (!)
- ideally: probability 1 for correct word; 0 for rest
- SGD with softmax output minimizes cross-entropy (and hence perplexity) of neural network

Feedforward neural language model: math



[Vaswani et al., 2013]

$$h_1 = \varphi W_1(Ex_1, Ex_2)$$
$$y = \text{softmax}(W_2 h_1)$$

Feedforward neural language model in SMT

FFNLM

- can be integrated as a feature in the log-linear SMT model [Schwenk et al., 2006]
- costly due to matrix multiplications and softmax
- solutions:
  - n-best reranking
  - variants of softmax (hierarchical softmax, self-normalization [NCE])
  - shallow networks; premultiplication of hidden layer
- scales well to many input words
  $\rightarrow$ models with source context [Devlin et al., 2014]

Recurrent neural network language model (RNNLM)

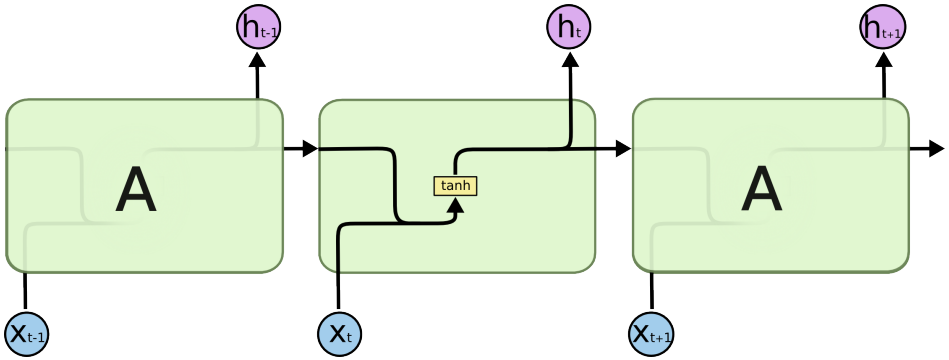
RNNLM [Mikolov et al., 2010]

- motivation: condition on arbitrarily long context
  $\rightarrow$ no Markov assumption
- we read in one word at a time, and update hidden state incrementally
- hidden state is initialized as empty vector at time step 0
- parameters:
  - embedding matrix E
  - feedforward matrices W_1, W_2
  - recurrent matrix U

$$h_i = \begin{cases} 0, & \text{, if } i = 0 \\ \tanh(W_1 E x_i + U h_{i-1}) & \text{, if } i > 0 \end{cases}$$

$$y_i = \text{softmax}(W_2 h_{i-1})$$

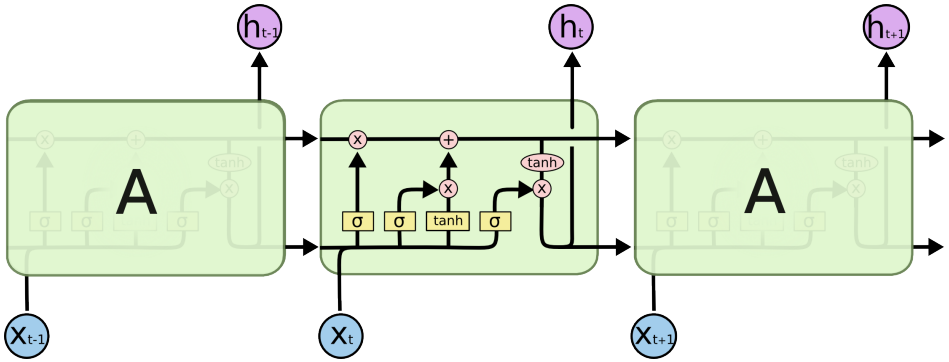
RNN variants



gated units

- alternative to plain RNN
- sigmoid layers σ act as “gates” that control flow of information
- allows passing of information over long time
  $\rightarrow$ avoids vanishing gradient problem
- strong empirical results
- popular variants:
  - Long Short Term Memory (LSTM) (shown)
  - Gated Recurrent Unit (GRU)

RNN variants



gated units

- alternative to plain RNN
- sigmoid layers σ act as “gates” that control flow of information
- allows passing of information over long time
  $\rightarrow$ avoids vanishing gradient problem
- strong empirical results
- popular variants:
  - Long Short Term Memory (LSTM) (shown)
  - Gated Recurrent Unit (GRU)

Modelling Translation

- Suppose that we have:
  - a source sentence S of length m $(x_1, \ldots, x_m)$
  - a target sentence T of length n $(y_1, \ldots, y_n)$
- We can express translation as a probabilistic model

$$T^* = \arg\max_T p(T|S)$$

- Expanding using the chain rule gives

$$\begin{aligned} p(T|S) &= p(y_1, \ldots, y_n | x_1, \ldots, x_m) \\ &= \prod_{i=1}^{n} p(y_i | y_1, \ldots, y_{i-1}, x_1, \ldots, x_m) \end{aligned}$$

Differences Between Translation and Language Model

- Target-side language model:

$$p(T) = \prod_{i=1}^{n} p(y_i|y_1, \ldots, y_{i-1})$$

- Translation model:

$$p(T|S) = \prod_{i=1}^{n} p(y_i|y_1, \ldots, y_{i-1}, x_1, \ldots, x_m)$$

- We could just treat sentence pair as one long sequence, but:
  - We do not care about $p(S)$
  - We may want different vocabulary, network architecture for source text

Differences Between Translation and Language Model

- Target-side language model:

$$p(T) = \prod_{i=1}^{n} p(y_i|y_1, \ldots, y_{i-1})$$

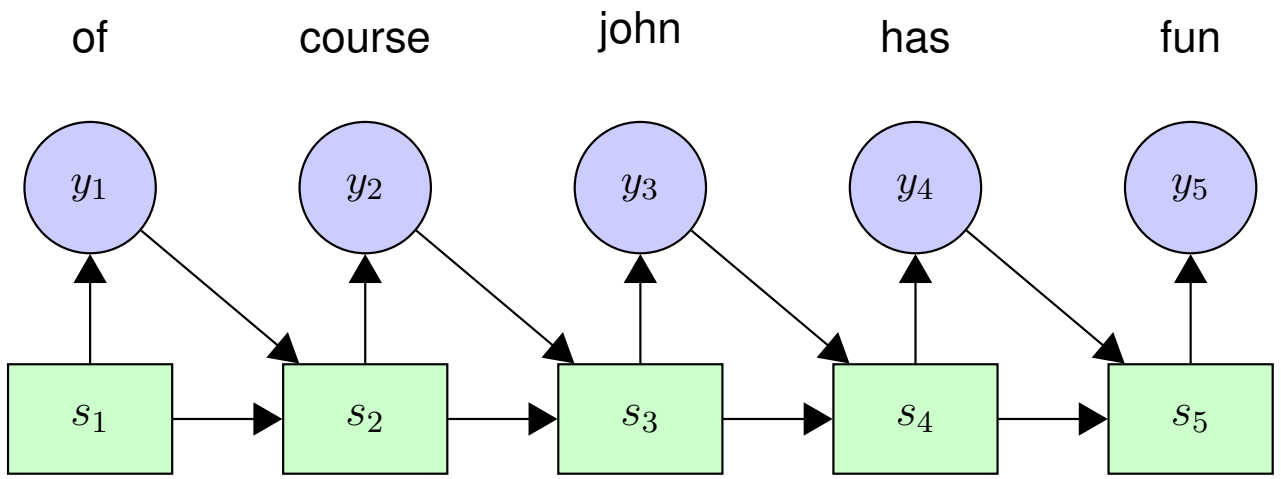
- Translation model:

$$p(T|S) = \prod_{i=1}^{n} p(y_i|y_1, \ldots, y_{i-1}, x_1, \ldots, x_m)$$

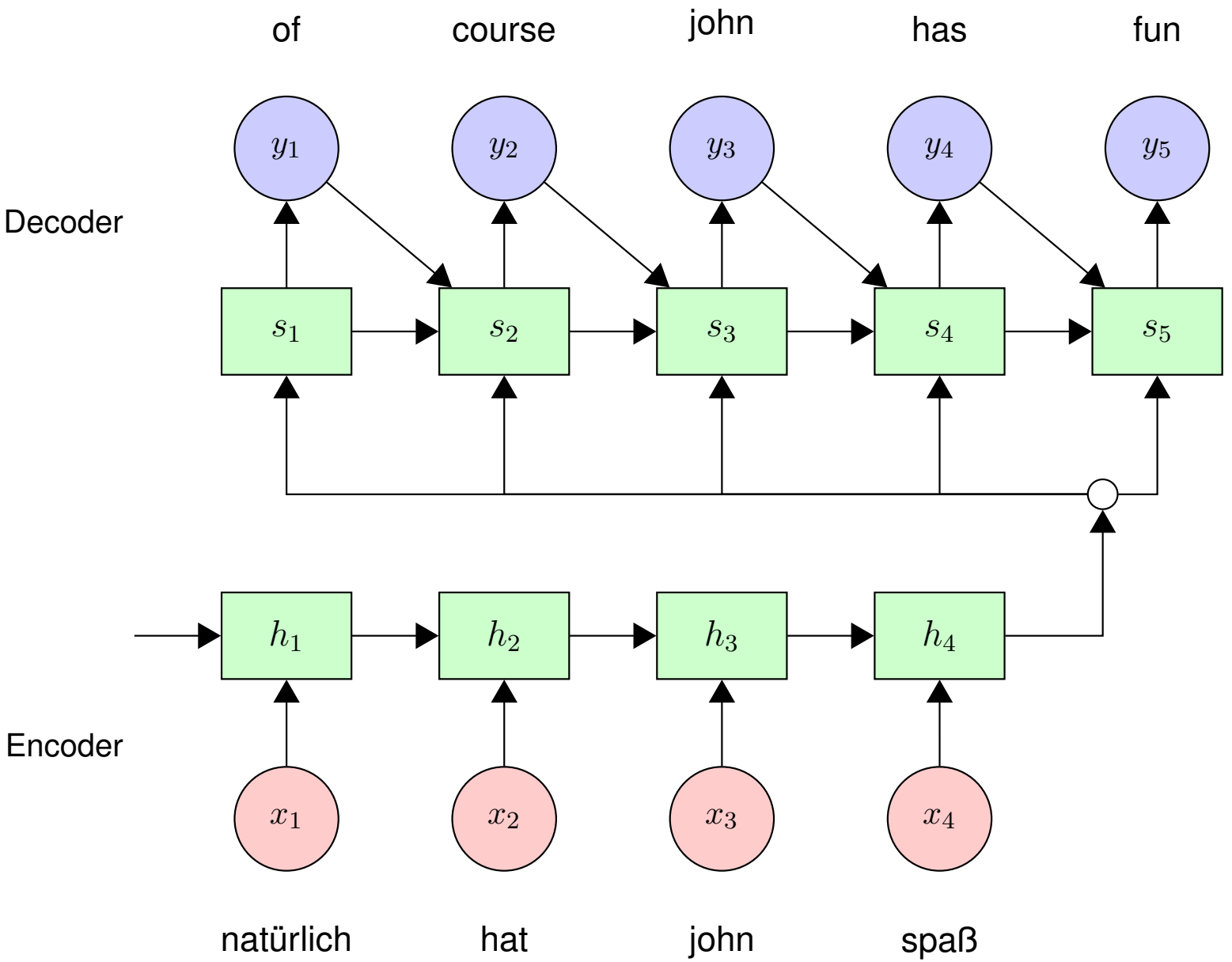
- We could just treat sentence pair as one long sequence, but:
  - We do not care about $p(S)$
  - We may want different vocabulary, network architecture for source text

$\rightarrow$ Use separate RNNs for source and target.

Encoder-Decoder for Translation

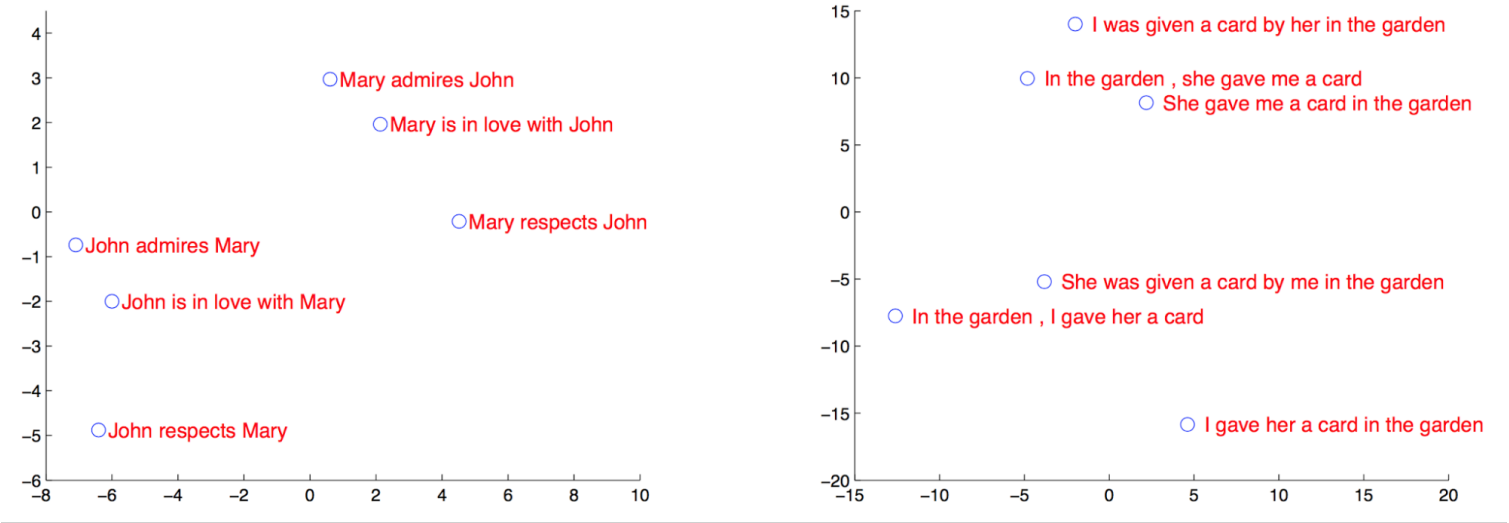


Encoder-Decoder for Translation



Summary vector

- Last encoder hidden-state “summarises” source sentence
- With multilingual training, we can potentially learn language-independent meaning representation

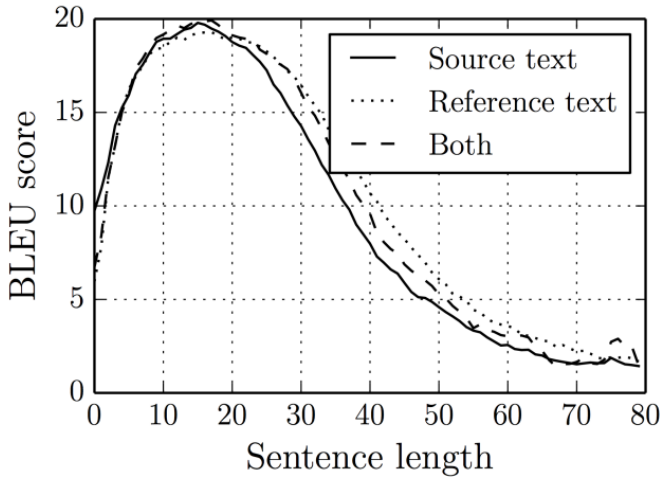


[Sutskever et al., 2014]

Summary vector as information bottleneck

Problem: Sentence Length

- Fixed sized representation degrades as sentence length increases
- Reversing source brings some improvement [Sutskever et al., 2014]

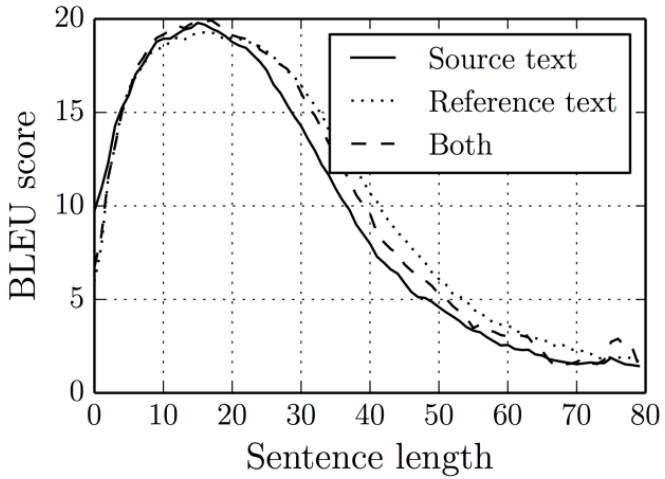


[Cho et al., 2014]

Summary vector as information bottleneck

Problem: Sentence Length

- Fixed sized representation degrades as sentence length increases
- Reversing source brings some improvement [Sutskever et al., 2014]

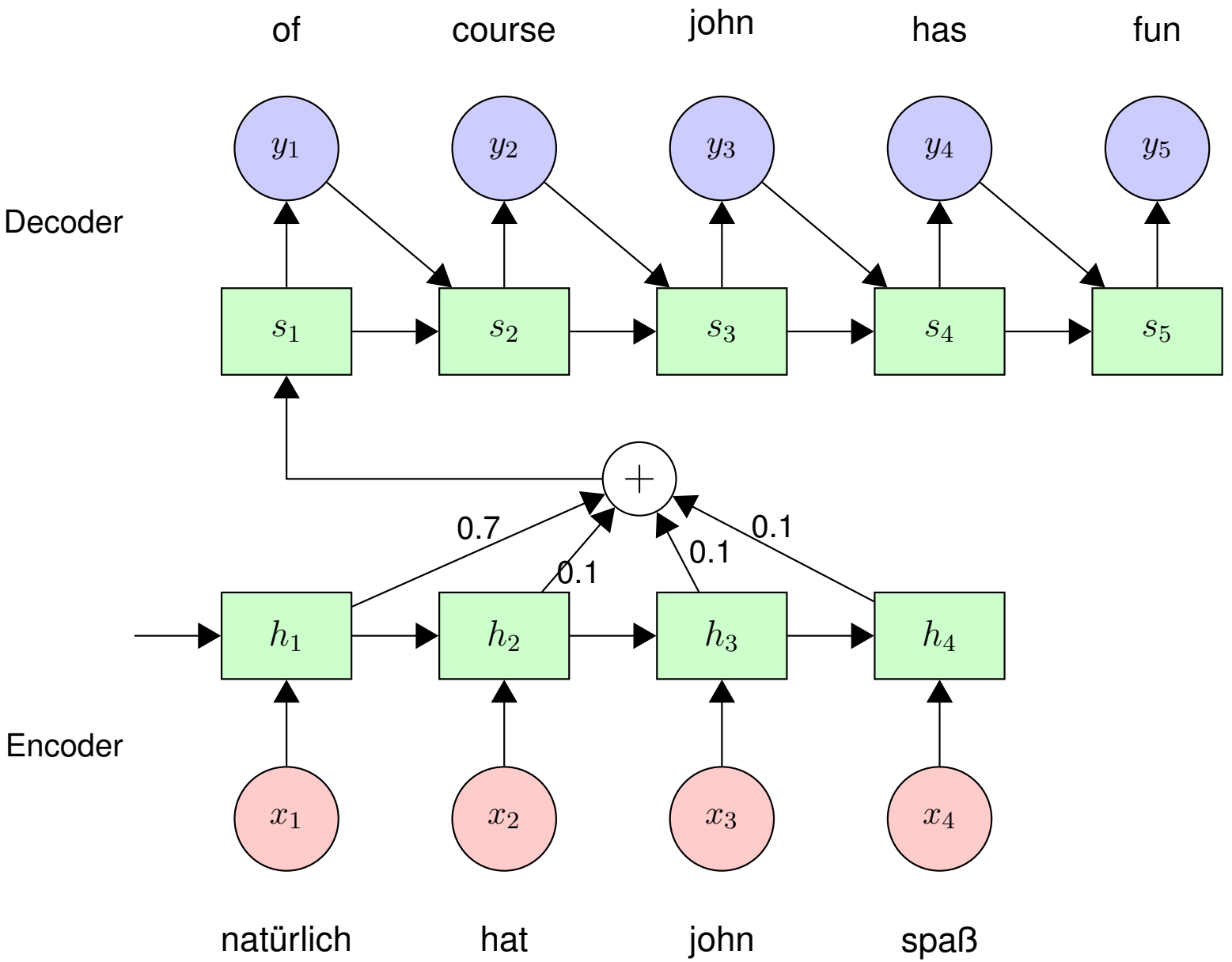


[Cho et al., 2014]

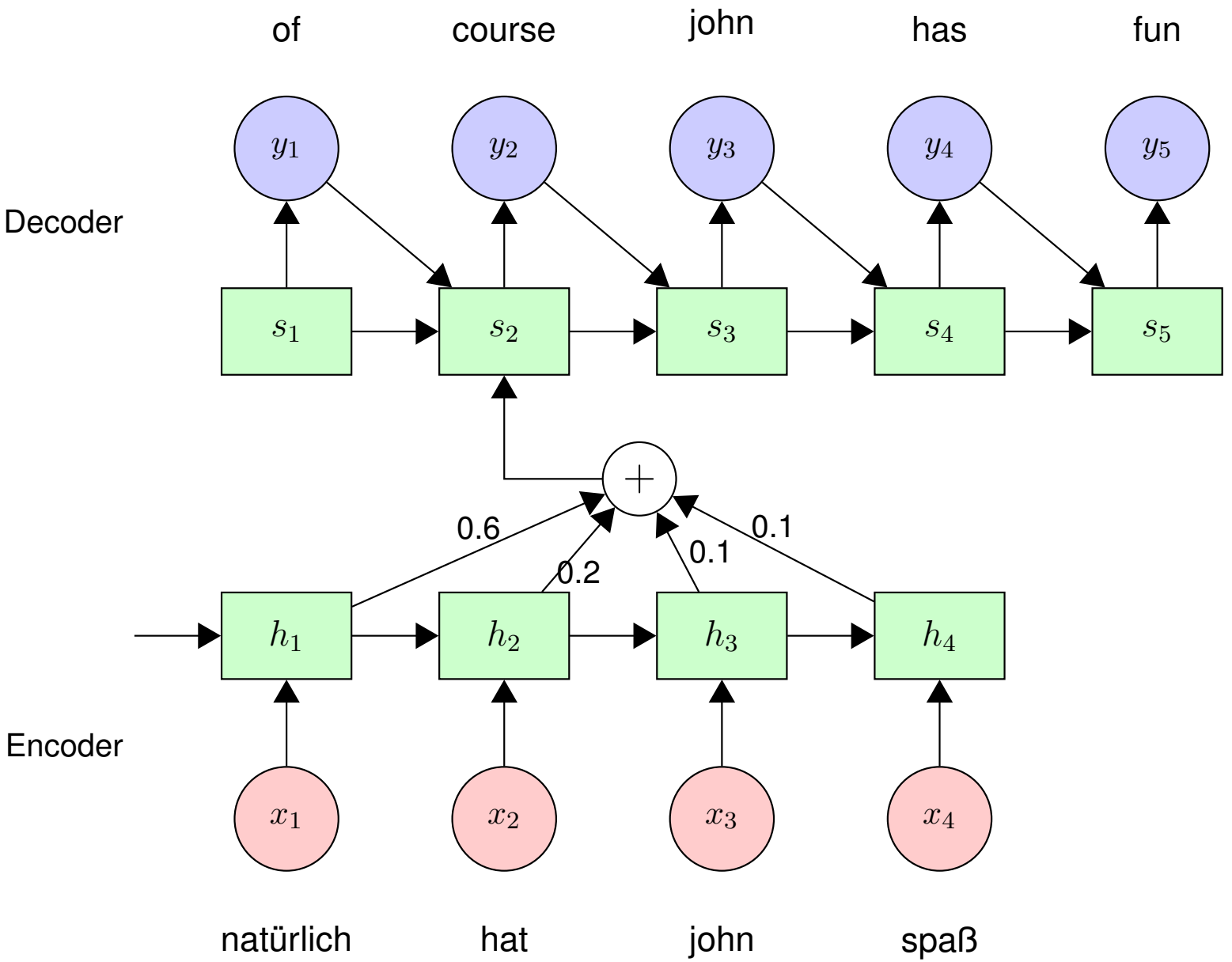
Solution: Attention

- Compute *context vector* as weighted average of source hidden states
- Weights computed by feed-forward network with softmax activation

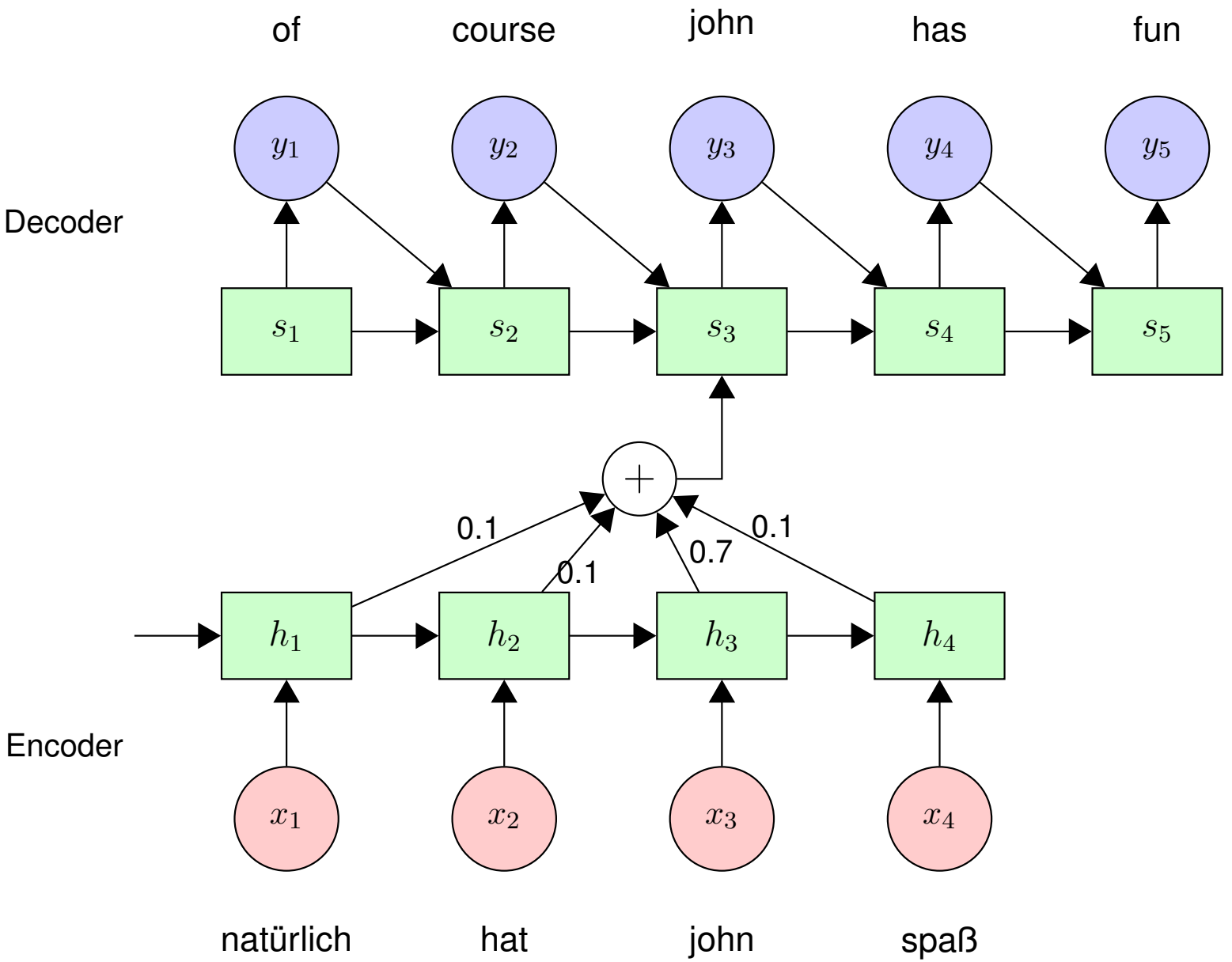
Encoder-Decoder with Attention



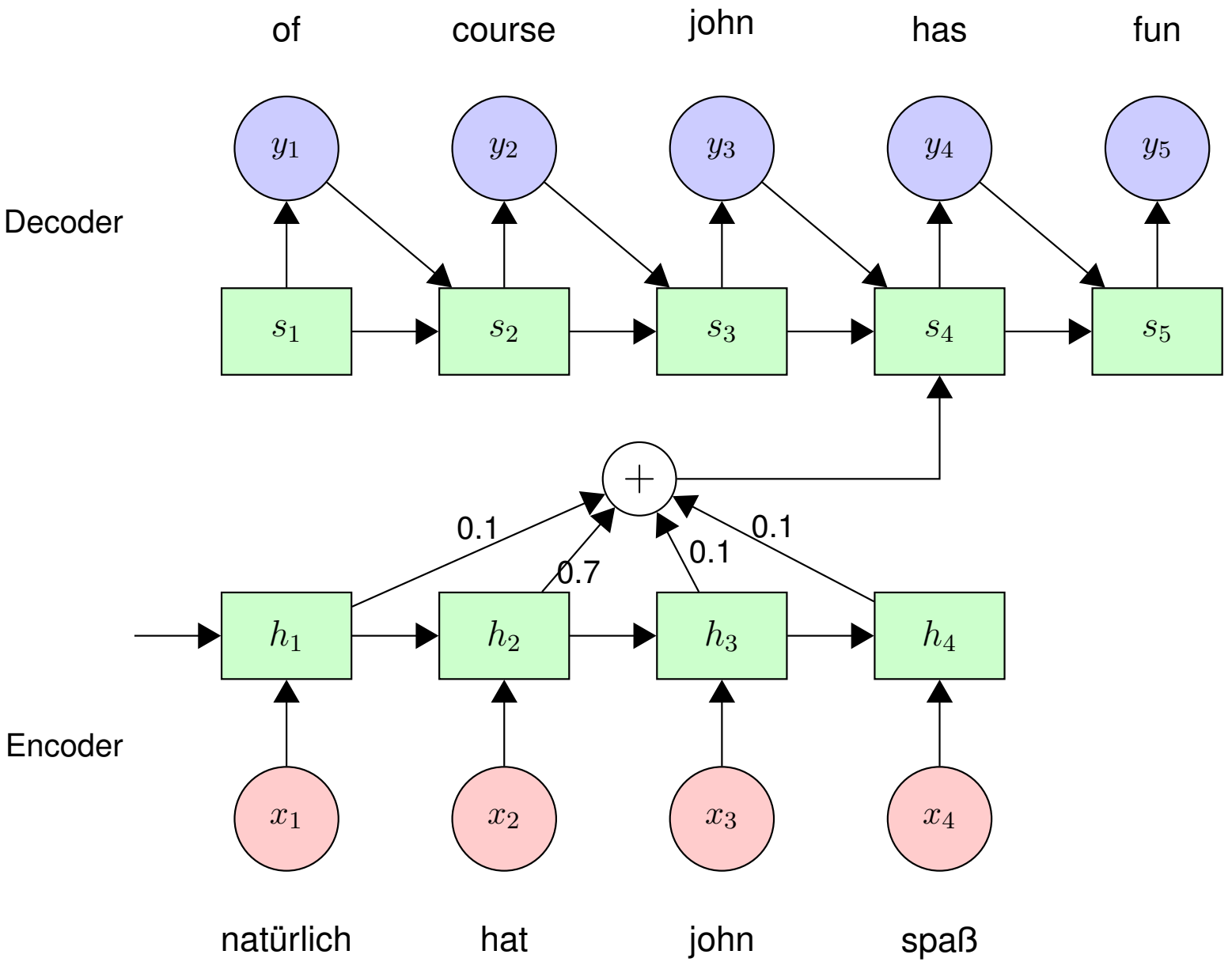
Encoder-Decoder with Attention



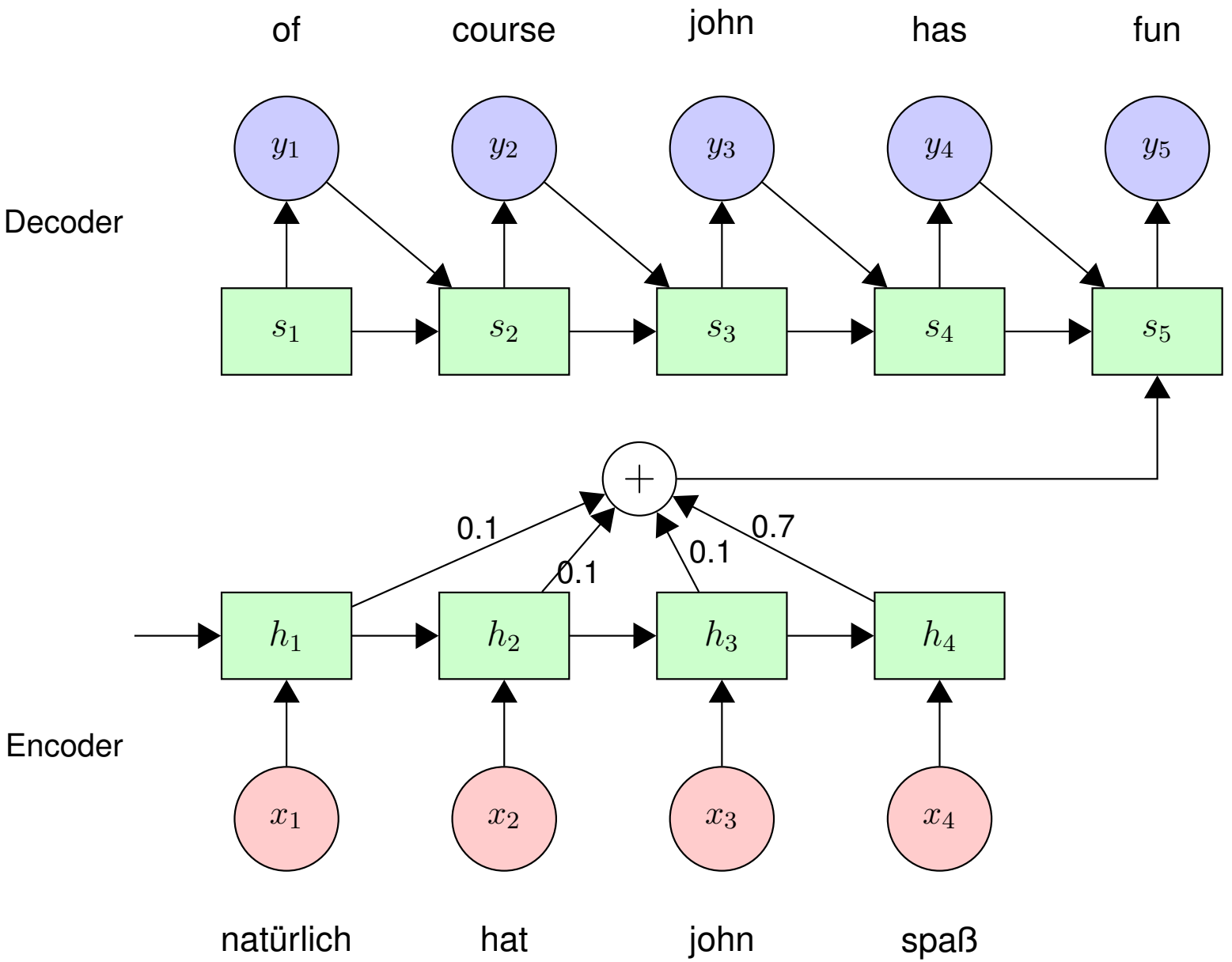
Encoder-Decoder with Attention



Encoder-Decoder with Attention



Encoder-Decoder with Attention



Attentional encoder-decoder: Maths

simplifications of model by [Bahdanau et al., 2015] (for illustration)

- plain RNN instead of GRU
- simpler output layer
- we do not show bias terms
- decoder follows *Look, Update, Generate* strategy [Sennrich et al., 2017]
- Details in `https://github.com/amunmt/amunmt/blob/master/contrib/notebooks/dl4mt.ipynb`

notation

- W, U, E, C, V are weight matrices (of different dimensionality)
  - E one-hot to embedding (e.g. $50000 \cdot 512$)
  - W embedding to hidden (e.g. $512 \cdot 1024$)
  - U hidden to hidden (e.g. $1024 \cdot 1024$)
  - C context (2x hidden) to hidden (e.g. $2048 \cdot 1024$)
  - V_o hidden to one-hot (e.g. $1024 \cdot 50000$)
- separate weight matrices for encoder and decoder (e.g. E_x and E_y)
- input X of length T_x; output Y of length T_y

Attentional encoder-decoder: Maths

encoder

$$\overrightarrow{h}_j = \begin{cases} 0, & \text{, if } j = 0 \\ \tanh(\overrightarrow{W}_x E_x x_j + \overrightarrow{U}_x h_{j-1}) & \text{, if } j > 0 \end{cases}$$

$$\overleftarrow{h}_j = \begin{cases} 0, & \text{, if } j = T_x + 1 \\ \tanh(\overleftarrow{W}_x E_x x_j + \overleftarrow{U}_x h_{j+1}) & \text{, if } j \leq T_x \end{cases}$$

$$h_j = (\overrightarrow{h}_j, \overleftarrow{h}_j)$$

Attentional encoder-decoder: Maths

decoder

$$s_i = \begin{cases} \tanh(W_s \overleftarrow{h}_i), & \text{, if } i = 0 \\ \tanh(W_y E_y y_{i-1} + U_y s_{i-1} + C c_i) & \text{, if } i > 0 \end{cases}$$

$$t_i = \tanh(U_o s_i + W_o E_y y_{i-1} + C_o c_i)$$

$$y_i = \text{softmax}(V_o t_i)$$

attention model

$$e_{ij} = v_a^\top \tanh(W_a s_{i-1} + U_a h_j)$$

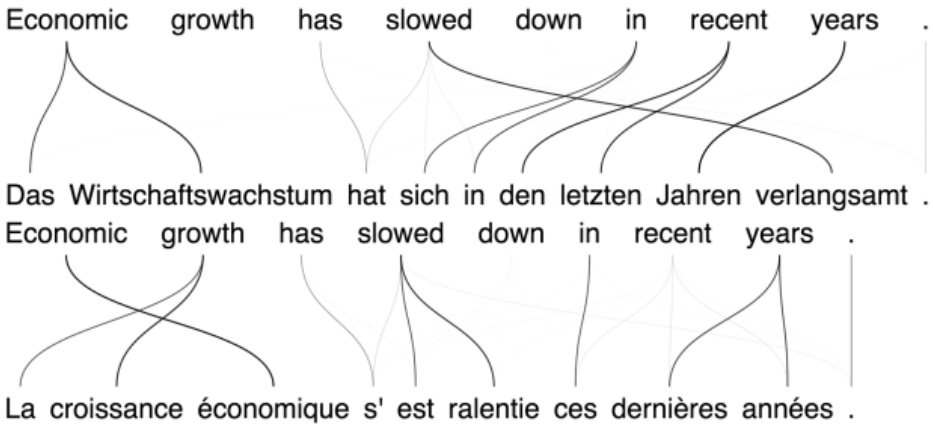
$$\alpha_{ij} = \text{softmax}(e_{ij})$$

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

Attention model

attention model

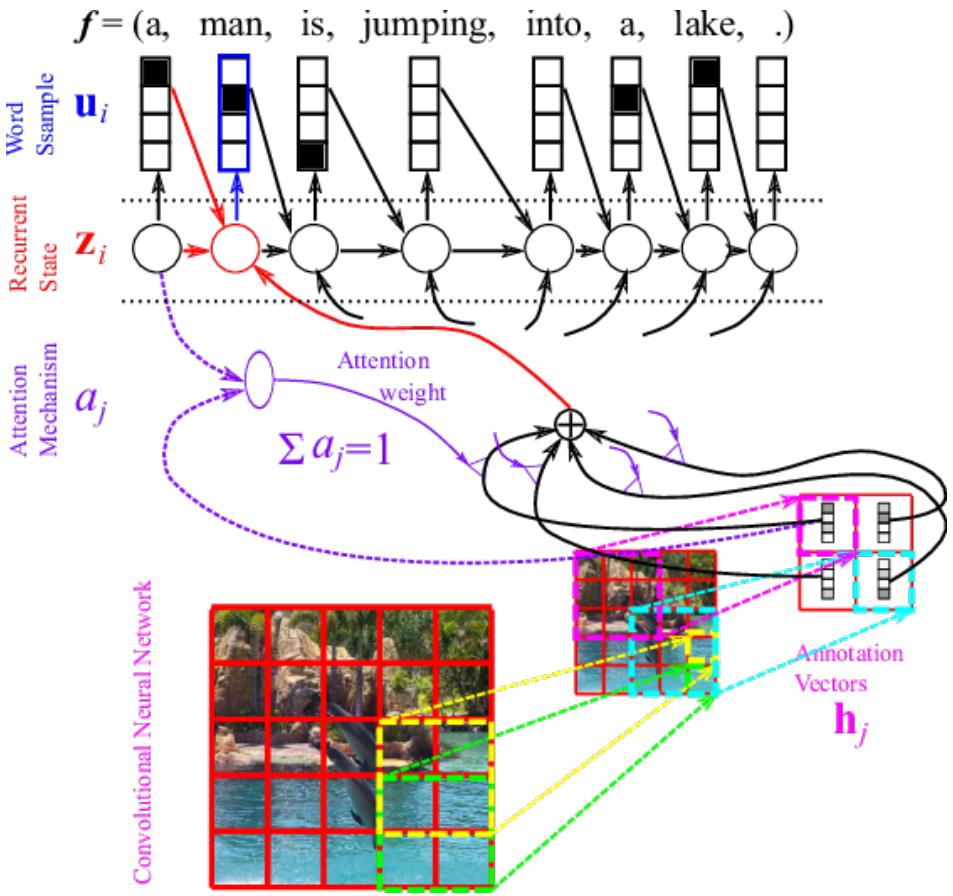
- side effect: we obtain alignment between source and target sentence
- information can also flow along recurrent connections, so there is no guarantee that attention corresponds to alignment
- applications:
  - visualisation
  - replace unknown words with back-off dictionary [Jean et al., 2015]
  - ...



Kyunghyun Cho
http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

Attention model

attention model also works with images:



[Cho et al., 2015]

Attention model

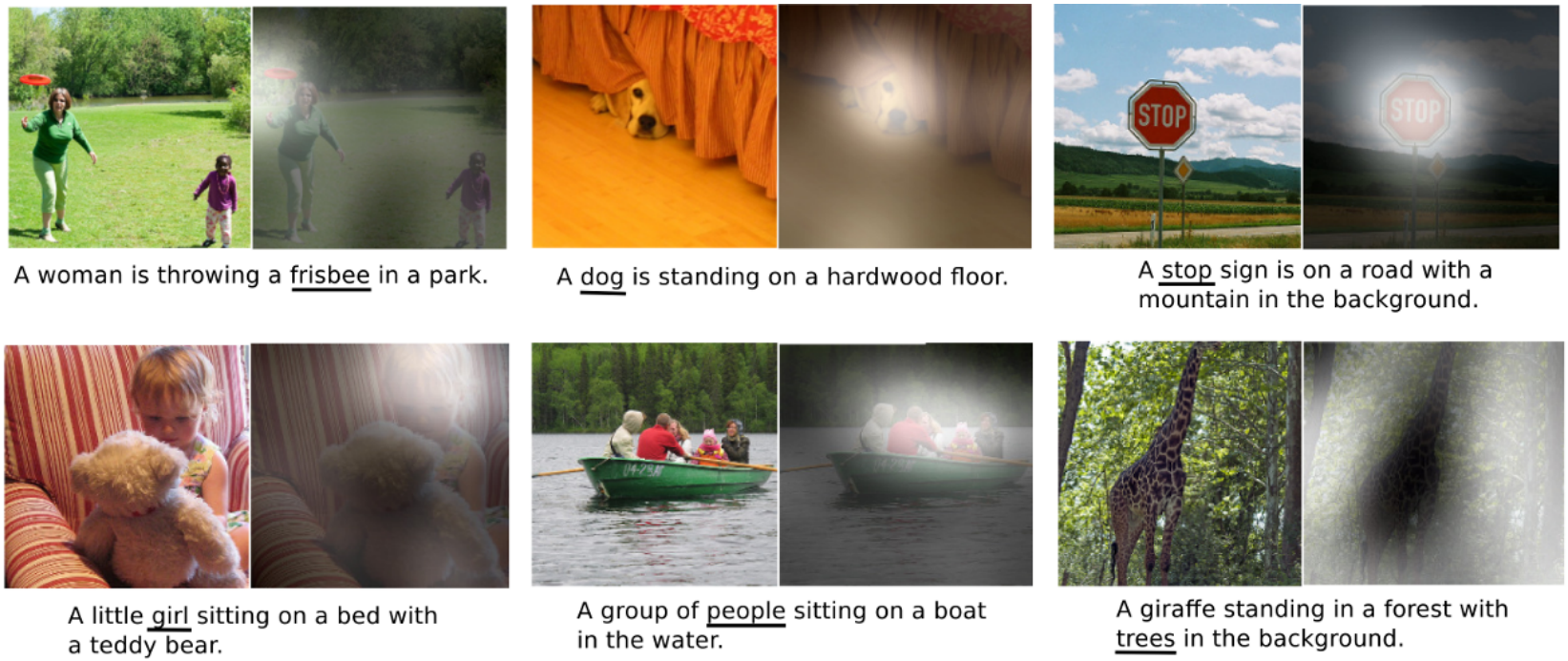


Fig. 5. Examples of the attention-based model attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word) [22]

[Cho et al., 2015]

Application of Encoder-Decoder Model

Scoring (a translation)

p(La, croissance, économique, s'est, ralentie, ces, dernières, années, . | Economic, growth, has, slowed, down, in, recent, year, .) = ?

Decoding (a source sentence)

Generate the most probable translation of a source sentence

$$y^* = \text{argmax}_y \, p(y|\text{Economic, growth, has, slowed, down, in, recent, year, .})$$

Decoding

exact search

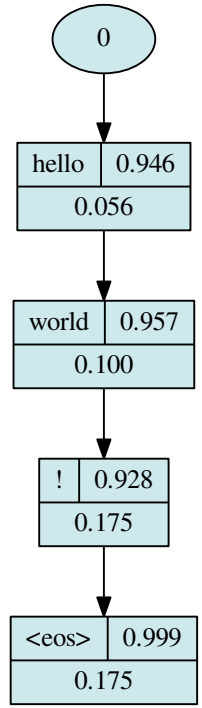
- generate every possible sentence T in target language
- compute score $p(T|S)$ for each
- pick best one

- intractable: $|\text{vocab}|^N$ translations for output length N
  $\rightarrow$ we need approximative search strategy

Decoding

approximative search/1: **greedy search**

- at each time step, compute probability distribution $P(y_i|S, y_{<i})$
- select y_i according to some heuristic:
  - sampling: sample from $P(y_i|S, y_{<i})$
  - greedy search: pick $\text{argmax}_y\, p(y_i|S, y_{<i})$
- continue until we generate *<eos>*



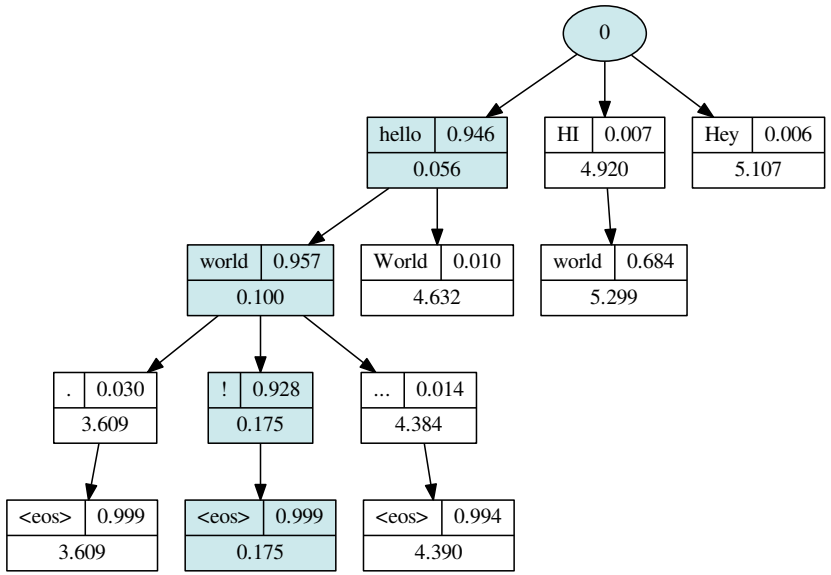
- efficient, but suboptimal

Decoding

approximative search/2: **beam search**

- maintain list of K hypotheses (beam)
- at each time step, expand each hypothesis k: $p(y_i^k|S, y_{<i}^k)$
- select K hypotheses with highest total probability:

$$\prod_i p(y_i^k|S, y_{<i}^k)$$



$K = 3$

- relatively efficient . . . beam expansion parallelisable
- currently default search strategy in neural machine translation
- small beam ($K \approx 10$) offers good speed-quality trade-off

Ensembles

- at each timestep, combine the probability distribution of M different ensemble components.
- combine operator: typically average (log-)probability

$$\log P(y_i|S, y_{<i}) = \frac{\sum_{m=1}^{M} \log P_m(y_i|S, y_{<i})}{M}$$

- requirements:
  - same output vocabulary
  - same factorization of Y
- internal network architecture may be different
- source representations may be different
  (extreme example: ensemble-like model with different source languages [Junczys-Dowmunt and Grundkiewicz, 2016])

Innovations in Edinburgh's WMT16 Systems

Basic encoder-decoder-with-attention, plus:

1. Subword models to allow translation of rare/unknown words
   → since networks have small, fixed vocabulary
2. Back-translated monolingual data as additional training data
   → allows us to make use of extensive monolingual resources
3. Combination of left-to-right and right-to-left models
   → Reduces "label-bias" problem
4. Bayesian dropout
   → Improves generalisation performance with small training data

Subwords for NMT: Motivation

MT is an open-vocabulary problem

- compounding and other productive morphological processes
  - they charge a carry-on bag fee.
  - sie erheben eine Hand|gepäck|gebühr.
- names
  - Obama(English; German)
  - Обама (Russian)
  - オバマ (o-ba-ma) (Japanese)
- technical terms, numbers, etc.

... but Neural MT architectures have small and fixed vocabulary

Subword units

segmentation algorithms: wishlist

- **open-vocabulary NMT**: encode *all* words through small vocabulary
- encoding generalizes to unseen words
- small text size
- good translation quality

our experiments

- after preliminary experiments, we use:
  - character n-grams (with shortlist of unsegmented words)
  - segmentation via *byte pair encoding*

Byte pair encoding for word segmentation

bottom-up character merging

- starting point: character-level representation
  → computationally expensive
- compress representation based on information theory
  → byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair ('A','B') with 'AB'
- hyperparameter: when to stop
  → controls vocabulary size

word	freq
'l o w </w>'	5
'l o w e r </w>'	2
'n e w e s t </w>'	6
'w i d e s t </w>'	3

vocabulary:
l o w </w> e r n s t i d

Byte pair encoding for word segmentation

bottom-up character merging

- starting point: character-level representation
  $\rightarrow$ computationally expensive
- compress representation based on information theory
  $\rightarrow$ byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair ('A','B') with 'AB'
- hyperparameter: when to stop
  $\rightarrow$ controls vocabulary size

word	freq
'l o w </w>'	5
'l o w e r </w>'	2
'n e w **es** t </w>'	6
'w i d **es** t </w>'	3

vocabulary:
l o w </w> e r n s t i d
es

Byte pair encoding for word segmentation

bottom-up character merging

- starting point: character-level representation
  $\rightarrow$ computationally expensive
- compress representation based on information theory
  $\rightarrow$ byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair ('A','B') with 'AB'
- hyperparameter: when to stop
  $\rightarrow$ controls vocabulary size

word	freq
'l o w </w>'	5
'l o w e r </w>'	2
'n e w **est** </w>'	6
'w i d **est** </w>'	3

vocabulary:
l o w </w> e r n s t i d
es **est**

Byte pair encoding for word segmentation

bottom-up character merging

- starting point: character-level representation
  $\rightarrow$ computationally expensive
- compress representation based on information theory
  $\rightarrow$ byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair ('A','B') with 'AB'
- hyperparameter: when to stop
  $\rightarrow$ controls vocabulary size

word	freq
'l o w </w>'	5
'l o w e r </w>'	2
'n e w **est</w>**'	6
'w i d **est</w>**'	3

vocabulary:
l o w </w> e r n s t i d
es est **est</w>**

Byte pair encoding for word segmentation

bottom-up character merging

- starting point: character-level representation
  $\rightarrow$ computationally expensive
- compress representation based on information theory
  $\rightarrow$ byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair ('A','B') with 'AB'
- hyperparameter: when to stop
  $\rightarrow$ controls vocabulary size

word	freq
'**lo** w </w>'	5
'**lo** w e r </w>'	2
'n e w est</w>'	6
'w i d est</w>'	3

vocabulary:
l o w </w> e r n s t i d
es est est</w> **lo**

Byte pair encoding for word segmentation

bottom-up character merging

- starting point: character-level representation
  $\rightarrow$ computationally expensive
- compress representation based on information theory
  $\rightarrow$ byte pair encoding [Gage, 1994]
- repeatedly replace most frequent symbol pair ('A','B') with 'AB'
- hyperparameter: when to stop
  $\rightarrow$ controls vocabulary size

word	freq
'**low** </w>'	5
'**low** e r </w>'	2
'n e w est</w>'	6
'w i d est</w>'	3

vocabulary:
l o w </w> e r n s t i d
es est est</w> lo **low**

Byte pair encoding for word segmentation

why BPE?

- open-vocabulary:
  operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency
  $\rightarrow$ trade-off between text length and vocabulary size

’l o w e s t </w>’

e s	$\rightarrow$	es
es t	$\rightarrow$	est
est </w>	$\rightarrow$	est</w>
l o	$\rightarrow$	lo
lo w	$\rightarrow$	low

Byte pair encoding for word segmentation

why BPE?

- open-vocabulary:
  operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency
  $\rightarrow$ trade-off between text length and vocabulary size

'l o w **es** t </w>'

e s	$\rightarrow$	**es**
es t	$\rightarrow$	est
est </w>	$\rightarrow$	est</w>
l o	$\rightarrow$	lo
lo w	$\rightarrow$	low

Byte pair encoding for word segmentation

why BPE?

- open-vocabulary:
  operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency
  $\rightarrow$ trade-off between text length and vocabulary size

'l o w **est** </w>'

e s	$\rightarrow$	es
es t	$\rightarrow$	**est**
est </w>	$\rightarrow$	est</w>
l o	$\rightarrow$	lo
lo w	$\rightarrow$	low

Byte pair encoding for word segmentation

why BPE?

- open-vocabulary:
  operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency
  $\rightarrow$ trade-off between text length and vocabulary size

'l o w **est</w>'

e s	$\rightarrow$	es
es t	$\rightarrow$	est
est </w>	$\rightarrow$	**est</w>**
l o	$\rightarrow$	lo
lo w	$\rightarrow$	low

Byte pair encoding for word segmentation

why BPE?

- open-vocabulary:
  operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency
  → trade-off between text length and vocabulary size

'**lo** w est</w>'

e s	→	es
es t	→	est
est </w>	→	est</w>
l o	→	**lo**
lo w	→	low

Byte pair encoding for word segmentation

why BPE?

- open-vocabulary:
  operations learned on training set can be applied to unknown words
- compression of frequent character sequences improves efficiency
  $\rightarrow$ trade-off between text length and vocabulary size

'**low** est</w>'

e s	$\rightarrow$	es
es t	$\rightarrow$	est
est </w>	$\rightarrow$	est</w>
l o	$\rightarrow$	lo
lo w	$\rightarrow$	**low**

Evaluation: data and methods

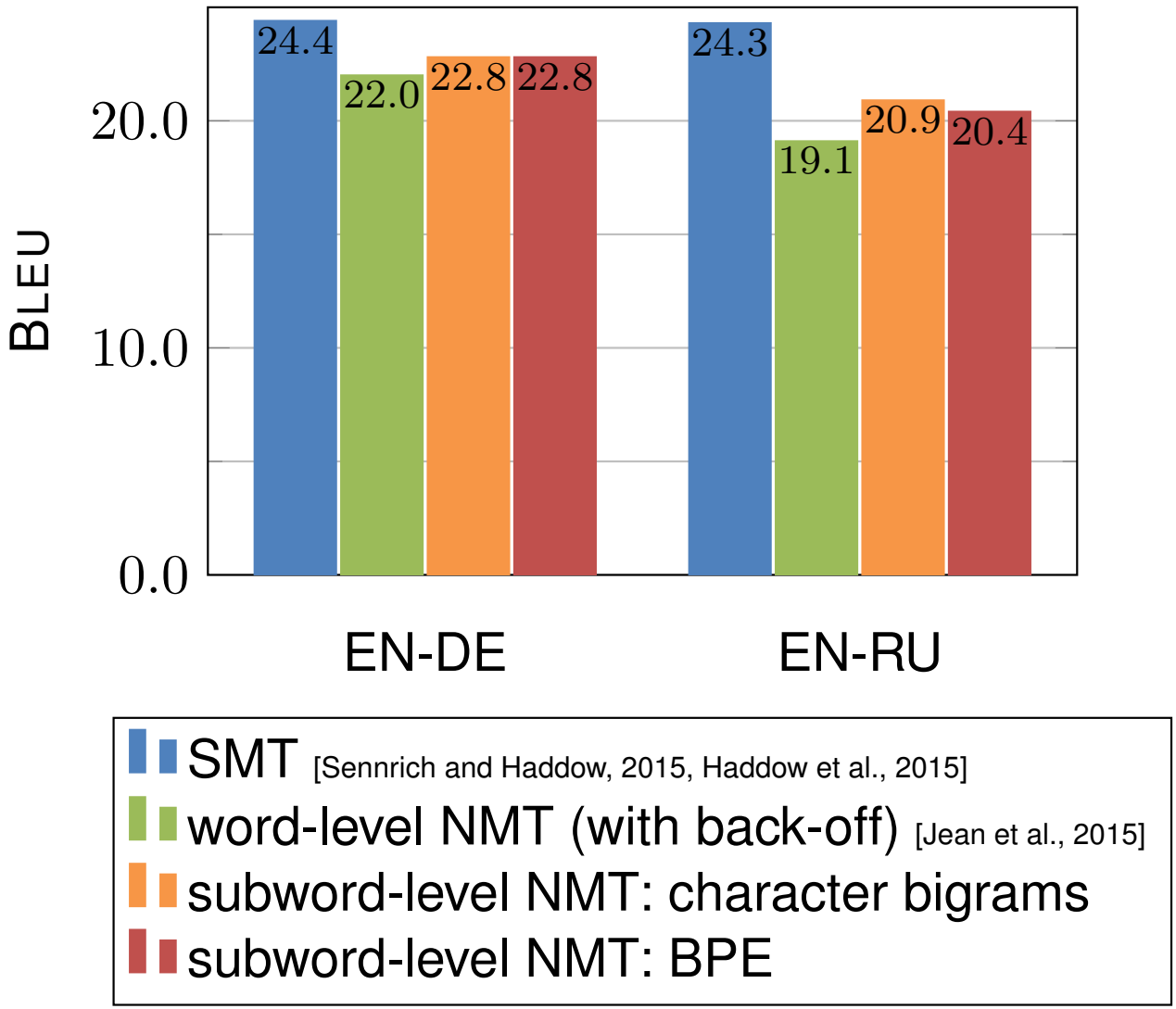
data

- WMT 15 English→German and English→Russian

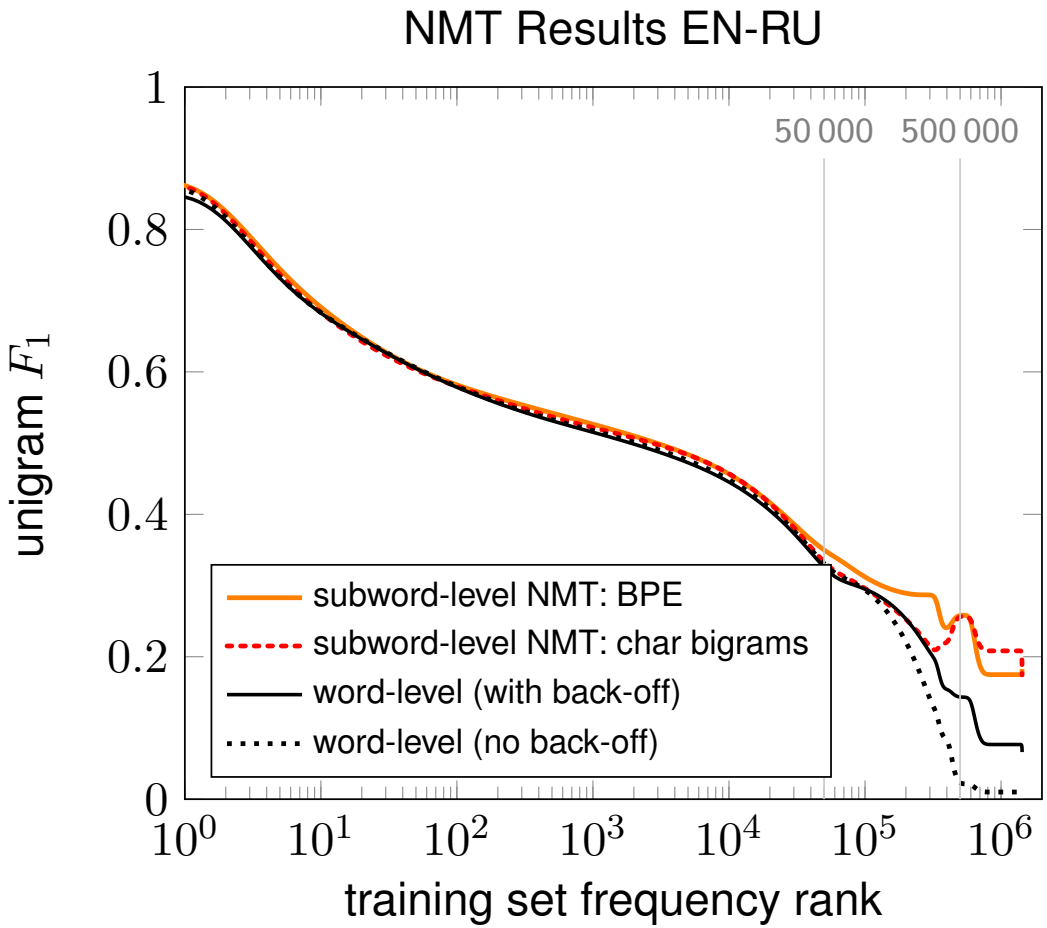
model

- attentional encoder–decoder neural network
- parameters and settings as in [Bahdanau et al, 2014]

Subword NMT: Translation Quality



Subword NMT: Translation Quality



Examples

system	sentence
source	health research institutes
reference	Gesundheitsforschungsinstitute
word-level (with back-off)	Forschungsinstitute
character bigrams	Fo|rs|ch|un|gs|in|st|it|ut|io|ne|n
BPE	Gesundheits|forsch|ungsin|stitute
source	rakfisk
reference	ракфиска (rakfiska)
word-level (with back-off)	rakfisk → UNK → rakfisk
character bigrams	ra|kf|is|k → ра|кф|ис|к (ra|kf|is|k)
BPE	rak|f|isk → рак|ф|иска (rak|f|iska)

BPE in WMT16 Systems

- Used **Joint** BPE
  - Just concatenate source and target, then train
  - Named-entities are split consistently
- Learn 89,500 merge operations
- Use ISO-9 transliteration for Russian:
  - transliterate Russian corpus into Latin script
  - learn BPE operations on concatenation of English and transliterated Russian corpus
  - transliterate BPE operations into Cyrillic
  - for Russian, apply both Cyrillic and Latin BPE operations
    $\rightarrow$ concatenate BPE files
- Set vocabulary size according to BPE vocabulary

Code available: `https://github.com/rsennrich/subword-nmt`

Monolingual Data in NMT

Why Monolingual Data for Phrase-based SMT?

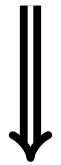
- more training data ✓
- relax independence assumptions ✓
- more appropriate training data (domain adaptation) ✓

Why Monolingual Data for NMT?

- more training data ✓
- relax independence assumptions ✗
- more appropriate training data (domain adaptation) ✓

Monolingual Data in NMT

encoder-decoder already conditions on previous target words



no architecture change required to learn from monolingual data

Monolingual Training Instances

Output prediction

- $p(y_i)$ is a function of hidden state s_i, previous output y_{i-1}, and source context vector c_i
- only difference to monolingual RNN: c_i

Problem

we have no source context c_i for monolingual training instances

Monolingual Training Instances

Output prediction

- $p(y_i)$ is a function of hidden state s_i, previous output y_{i-1}, and source context vector c_i
- only difference to monolingual RNN: c_i

Problem

we have no source context c_i for monolingual training instances

Solutions

- two methods to deal with missing source context:
  - empty/dummy source context c_i
    $\rightarrow$ danger of unlearning conditioning on source
  - produce synthetic source sentence via back-translation
    $\rightarrow$ get approximation of c_i

Monolingual Training Instances

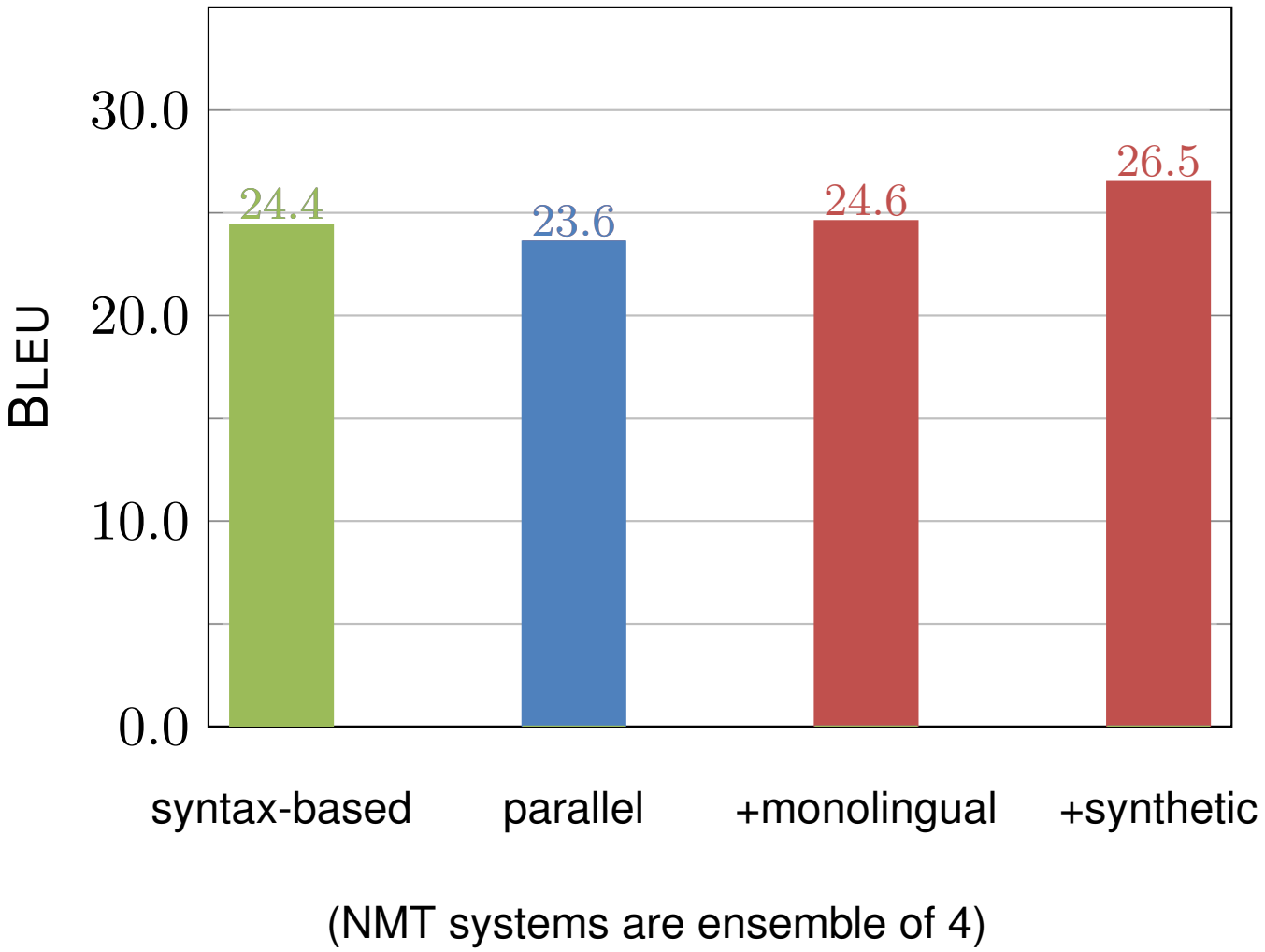
Dummy source

- 1-1 mix of parallel and monolingual training instances
- randomly sample from monolingual data each epoch
- freeze encoder/attention layers for monolingual training instances

Synthetic source

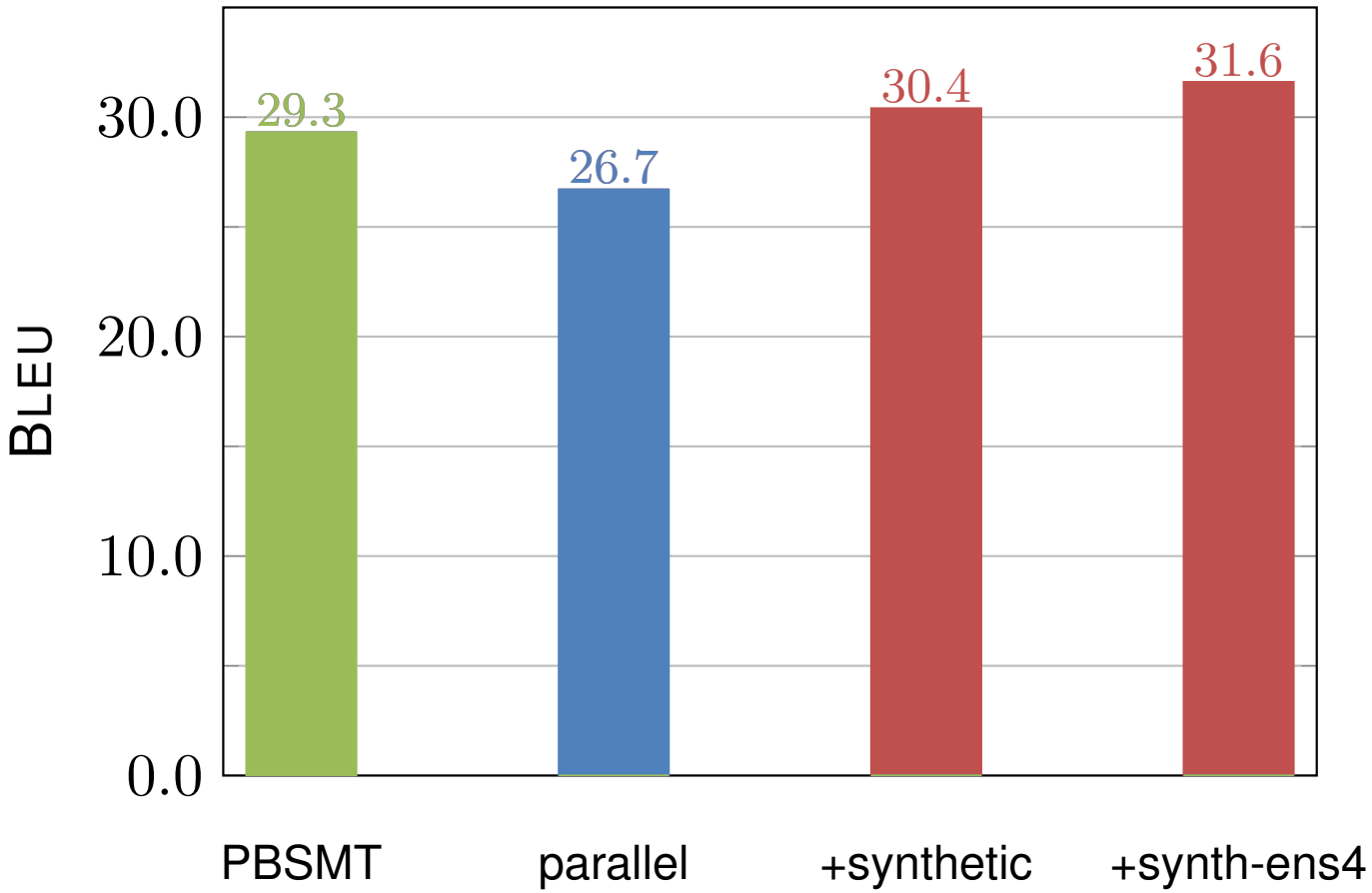
- 1-1 mix of parallel and monolingual training instances
- randomly sample from back-translated data
- training does not distinguish between real and synthetic parallel data

Evaluation: WMT 15 English→German



(NMT systems are ensemble of 4)

Evaluation: WMT 15 German→English



Why is monolingual data helpful?

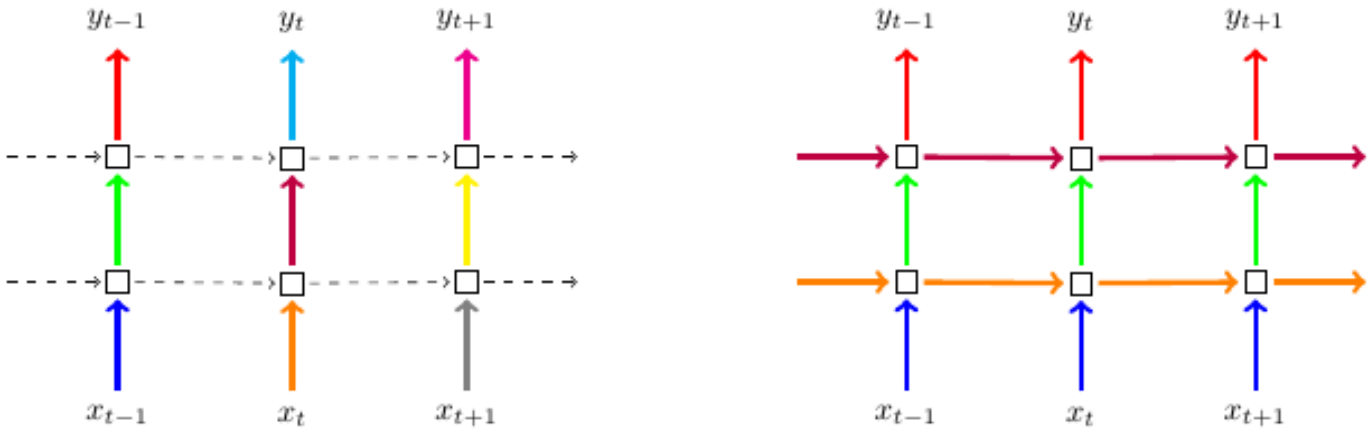
- Domain adaptation effect
- Reduces over-fitting
- Improves fluency

(See [Sennrich et al., 2016] for more analysis.)

Left-to-Right / Right-to-Left Reranking

- Target history is strong signal for next prediction
  - History is reliable at training time, but not at test time
  - Low-entropy output words lead to poor translation
  - Similar to **label bias** problem
- Reranking with reverse model can help
  1. Train two models, one has target reversed
  2. Generate n-best lists with one model
  3. Rescore lists with second model
  4. Rerank using combined scores
- Consistent increase (0.5 – 1) in BLEU

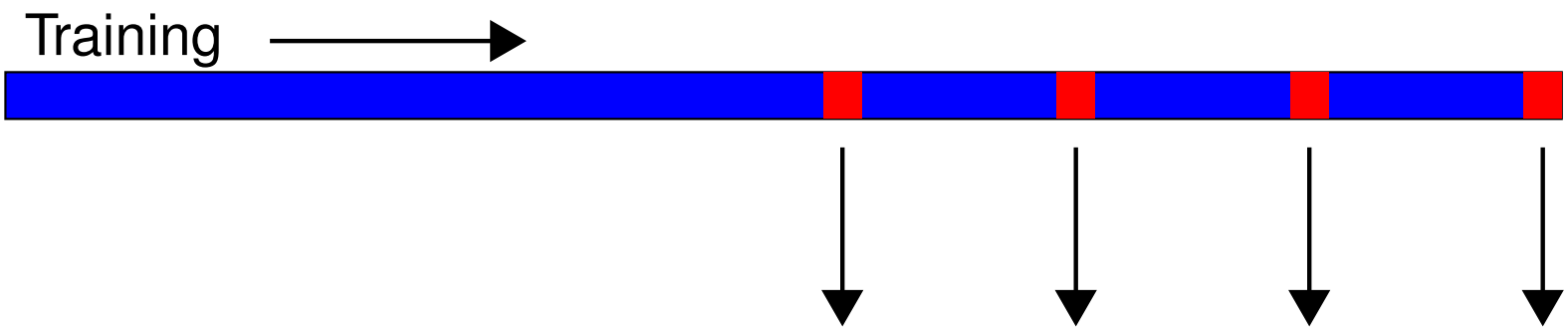
Bayesian Dropout



[Gal, 2015]

- Dropout (randomly zeroing activations in training) prevents overfitting
- Follow [Gal, 2015] and repeat mask across timesteps
- Necessary for English↔Romanian (0.6M sentences)
- Masks of 0.1-0.2 provide gain of 4-5 BLEU

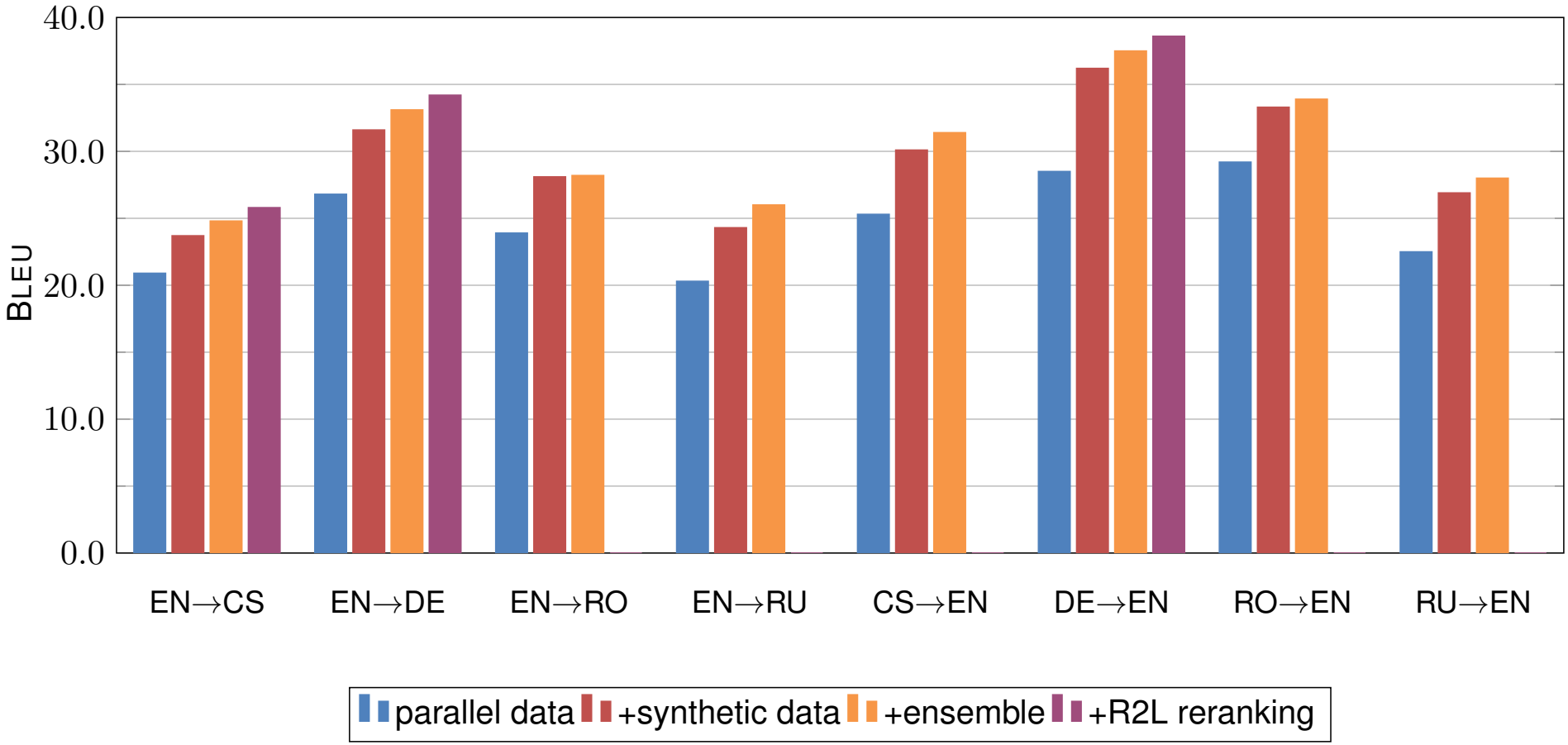
Checkpoint Ensembling



$$p(e|f) \quad = \quad p_1(e|f) \times p_2(e|f) \times p_3(e|f) \times p_4(e|f)$$

- Ensembling improves **performance** and **stability**
- Checkpoint ensembling much cheaper than independent runs

Putting it all together: WMT16 Results



Comparison between phrase-based and neural MT

human analysis of NMT (reranking) [Neubig et al., 2015]

- NMT is more grammatical
  - word order
  - insertion/deletion of function words
  - morphological agreement
- minor degradation in lexical choice?

Comparison between phrase-based and neural MT

analysis of IWSLT 2015 results [Bentivogli et al., 2016]

- human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

system	HTER (no *shift*)			HTER
	word	lemma	%Δ	(*shift* only)
PBSMT [Ha et al., 2015]	28.3	23.2	-18.0	3.5
NMT [Luong and Manning, 2015]	21.7	18.7	-13.7	1.5

- word-level is closer to lemma-level performance: better at inflection/agreement
- improvement on lemma-level: better lexical choice
- fewer shift errors: better word order

Comparison between phrase-based and neural MT

analysis of IWSLT 2015 results [Bentivogli et al., 2016]

- human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

system	HTER (no *shift*)			HTER
	word	lemma	%Δ	(*shift* only)
PBSMT [Ha et al., 2015]	28.3	23.2	-18.0	3.5
NMT [Luong and Manning, 2015]	21.7	18.7	-13.7	1.5

- word-level is closer to lemma-level performance: better at inflection/agreement
- improvement on lemma-level: better lexical choice
- fewer shift errors: better word order

Comparison between phrase-based and neural MT

analysis of IWSLT 2015 results [Bentivogli et al., 2016]

- human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

system	HTER (no *shift*)			HTER
	word	lemma	%Δ	(*shift* only)
PBSMT [Ha et al., 2015]	28.3	23.2	-18.0	3.5
NMT [Luong and Manning, 2015]	21.7	18.7	-13.7	1.5

- word-level is closer to lemma-level performance: better at inflection/agreement
- improvement on lemma-level: better lexical choice
- fewer shift errors: better word order

Comparison between phrase-based and neural MT

analysis of IWSLT 2015 results [Bentivogli et al., 2016]

- human-targeted translation error rate (HTER) based on automatic translation and human post-edit
- 4 error types: substitution, insertion, deletion, shift

system	HTER (no *shift*)			HTER
	word	lemma	%Δ	(*shift* only)
PBSMT [Ha et al., 2015]	28.3	23.2	-18.0	3.5
NMT [Luong and Manning, 2015]	21.7	18.7	-13.7	1.5

- word-level is closer to lemma-level performance: better at inflection/agreement
- improvement on lemma-level: better lexical choice
- fewer shift errors: better word order

Adequacy vs. Fluency in WMT16 Evaluation

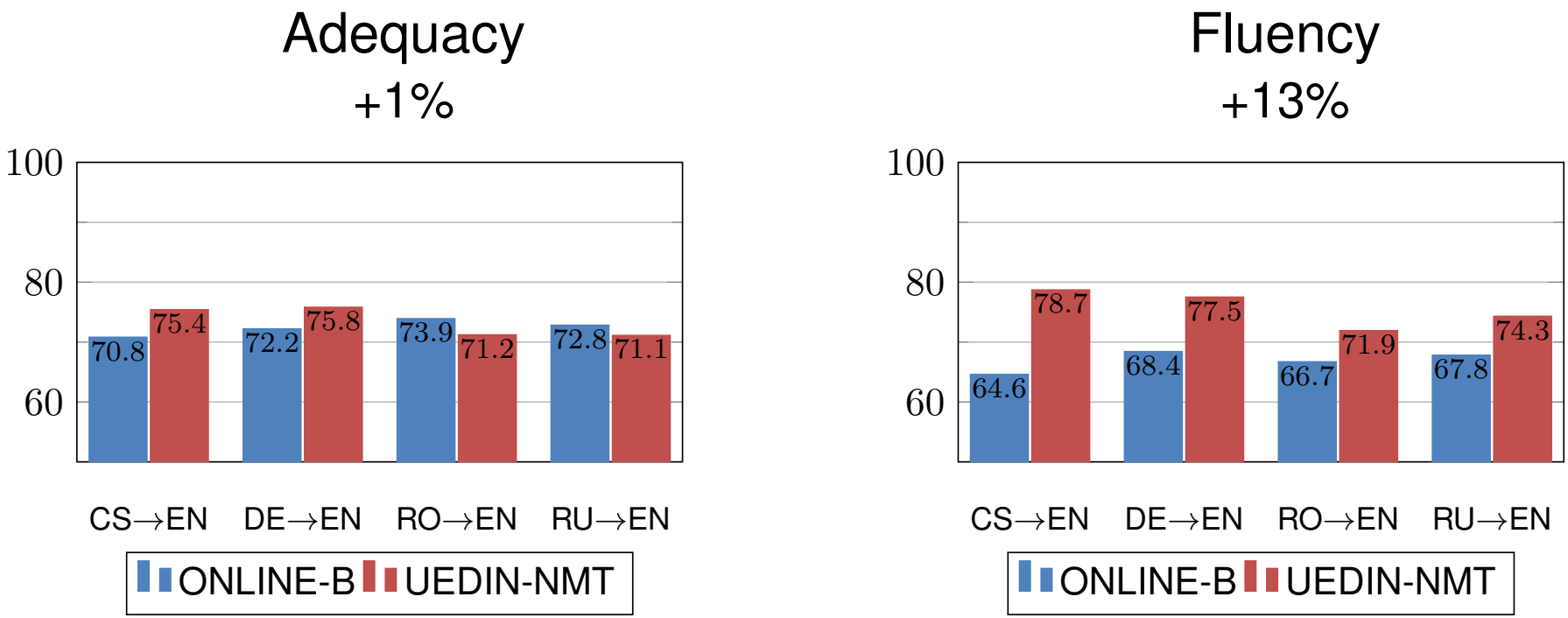


Figure : WMT16 direct assessment results

Human Evaluation in TraMOOC

- comparison of NMT and PBSMT for EN→{DE,EL,PT,RU}
- direct assessment:
  - NMT obtains higher fluency judgment than PBSMT: +10%
  - NMT only obtains small improvement in adequacy judgment: +1%
- post-editing:
  - NMT reduces technical effort (keystrokes): -13%
  - small reduction in post-editing time: -4%

→ NMT errors more difficult to identify

Error Annotation

category	SMT	NMT	difference
inflectional morphology	2274	1799	**-21%**
word order	1098	691	**-37%**
omission	421	362	-14%
addition	314	265	-16%
mistranslation	1593	1552	-3%
"no issue"	449	788	**+75%**

Assessing MT Quality with Contrastive Translation Pairs

Questions

- how well does NMT perform for specific linguistic phenomena?
- example: is grammaticality affected by choice of subword unit?

Method [Sennrich, 2017]

- compare probability of human reference translation with contrastive translation that introduces a specific type of error
  $\rightarrow$ NMT model should prefer reference
- errors related to:
  - morphosyntactic agreement
  - discontiguous units of meaning
  - polarity
  - transliteration

Contrastive Translation Pairs: Example

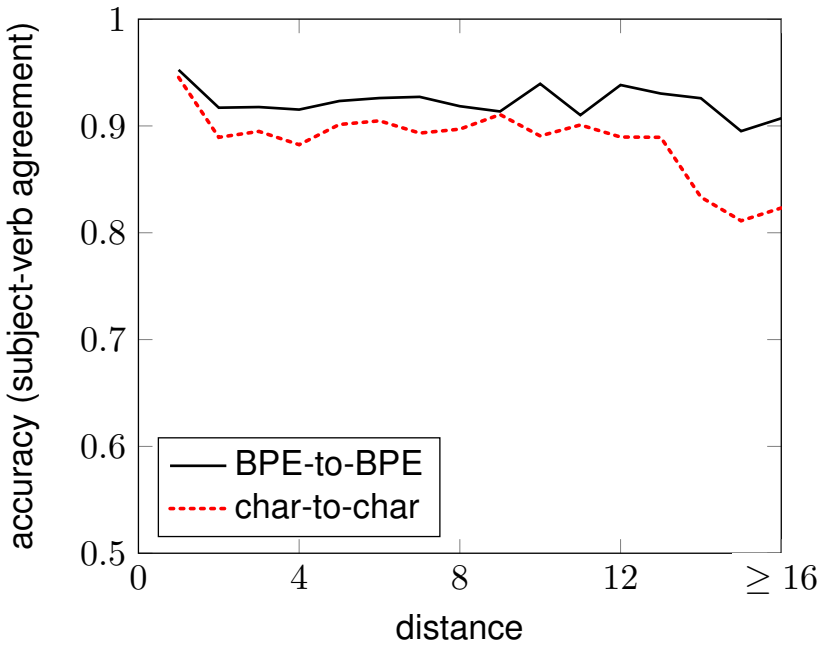
English	[...] that the **plan will** be approved
German (correct)	[...], dass der **Plan** verabschiedet **wird**
German (contrastive)	* [...], dass der **Plan** verabschiedet **werden**

subject-verb agreement

Assessing MT Quality with Contrastive Translation Pairs

Results

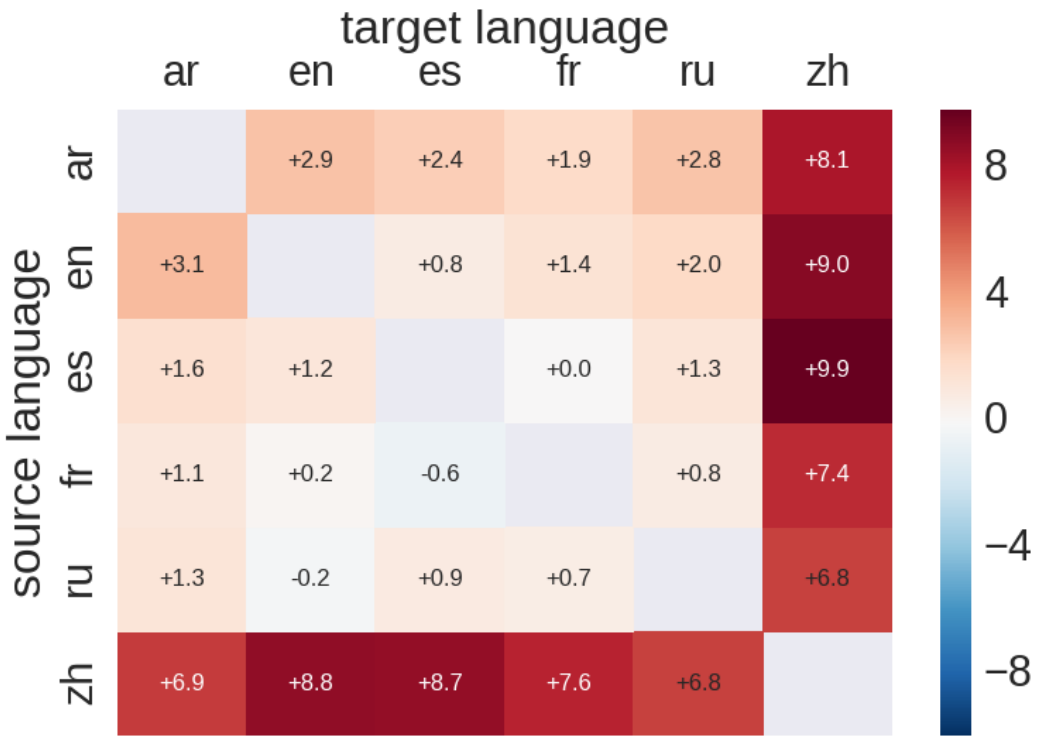
- WMT16 NMT system detects agreement errors with high accuracy – 96.6–98.7%.
- character-level system [Lee et al., 2016] better than BPE-to-BPE system at transliteration, but worse at morphosyntactic agreement
- difference higher for agreement over long distances



NMT vs. PBMT: An extended test [Junczys-Dowmunt et al., 2016a]

Experimental Setup

- Training and test drawn from UN corpus
  - Multi-parallel, 11M lines
  - Arabic, Chinese, English, French, Russian, Spanish
- Use only parallel data, evaluate with BLEU on 4000 sentences



Why is neural MT output more grammatical?

phrase-based SMT

- log-linear combination of many “weak” features
- data sparsenesss triggers back-off to smaller units
- strong independence assumptions

neural MT

- end-to-end trained model
- generalization via continuous space representation
- output conditioned on full source text and target history

Resource Usage

- We all want our experiments to finish faster . . .
- What influences training speed/memory usage?
  - Number of model parameters, especially vocabulary size
  - Size of training instance (max. length $\times$ batch size)
  - Hardware and library versions
- Decoding speed
  - Less important for NMT researchers
  - Standard Nematus model $\rightarrow$ Use AmuNMT (hand-crafted GPU code)

Hardware/Library Choice

Hardware	Theano	CuDNN	gpuarray	Sentence/s
CPU (Xeon E5-2680)	0.8.2	No	No	2.5
GPU (Titan X Pascal)	0.8.2	No	No	83
GPU (Titan X Pascal)	0.8.2	5.10	No	138
GPU (Titan X Pascal)	0.9b	5.10	No	171
GPU (Titan Black)	0.9b	5.10	No	109
GPU (Titan X)	0.9b	5.10	No	110
GPU (GTX 1080)	0.9b	5.10	No	177
GPU (Tesla M60)	0.9b	5.10	No	110
GPU (Titan X Pascal)	0.9rc3	5.10	Yes	227

Hyperparameters: Efficiency

- hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- memory consumption affected by
  - number of model parameters
  - size of training instance (length · batchsize)
- we show some profile output for guidance:
  - Nematus ('test_train.sh')
  - NVIDIA GTX 1080 GPU

layer size		vocabulary		batchsize	maxlen	GPU memory	speed	
embed	hidden	source	target			(peak)	sents/s	words/s
256	512	30000	30000	40	50	1.2 GB	174	4080
256	512	30000	60000	40	50	2.1 GB	148	3470
256	512	60000	60000	40	50	2.3 GB	145	3410
256	1024	60000	60000	40	50	2.7 GB	95	2220
512	1024	60000	60000	40	50	3.6 GB	79	1850
512	1024	60000	60000	80	50	4.9 GB	110	2600
512	1024	60000	60000	80	80	6.6 GB	87	2570

Hyperparameters: Efficiency

- hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- memory consumption affected by
  - number of model parameters
  - size of training instance (length · batchsize)
- we show some profile output for guidance:
  - Nematus ('test_train.sh')
  - NVIDIA GTX 1080 GPU

layer size		vocabulary		batchsize	maxlen	GPU memory	speed	
embed	hidden	source	target			(peak)	sents/s	words/s
256	512	30000	30000	40	50	1.2 GB	174	4080
256	512	30000	60000	40	50	2.1 GB	148	3470
256	512	60000	60000	40	50	2.3 GB	145	3410
256	1024	60000	60000	40	50	2.7 GB	95	2220
512	1024	60000	60000	40	50	3.6 GB	79	1850
512	1024	60000	60000	80	50	4.9 GB	110	2600
512	1024	60000	60000	80	80	6.6 GB	87	2570

Hyperparameters: Efficiency

- hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- memory consumption affected by
  - number of model parameters
  - size of training instance (length · batchsize)
- we show some profile output for guidance:
  - Nematus ('test_train.sh')
  - NVIDIA GTX 1080 GPU

layer size		vocabulary		batchsize	maxlen	GPU memory	speed	
embed	hidden	source	target			(peak)	sents/s	words/s
256	512	30000	30000	40	50	1.2 GB	174	4080
256	512	30000	60000	40	50	2.1 GB	148	3470
256	512	60000	60000	40	50	2.3 GB	145	3410
256	1024	60000	60000	40	50	2.7 GB	95	2220
512	1024	60000	60000	40	50	3.6 GB	79	1850
512	1024	60000	60000	80	50	4.9 GB	110	2600
512	1024	60000	60000	80	80	6.6 GB	87	2570

Hyperparameters: Efficiency

- hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- memory consumption affected by
  - number of model parameters
  - size of training instance (length · batchsize)
- we show some profile output for guidance:
  - Nematus ('test_train.sh')
  - NVIDIA GTX 1080 GPU

layer size		vocabulary		batchsize	maxlen	GPU memory	speed	
embed	hidden	source	target			(peak)	sents/s	words/s
256	512	30000	30000	40	50	1.2 GB	174	4080
256	512	30000	60000	40	50	2.1 GB	148	3470
256	512	60000	60000	40	50	2.3 GB	145	3410
256	1024	60000	60000	40	50	2.7 GB	95	2220
512	1024	60000	60000	40	50	3.6 GB	79	1850
512	1024	60000	60000	80	50	4.9 GB	110	2600
512	1024	60000	60000	80	80	6.6 GB	87	2570

Hyperparameters: Efficiency

- hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- memory consumption affected by
  - number of model parameters
  - size of training instance (length · batchsize)
- we show some profile output for guidance:
  - Nematus ('test_train.sh')
  - NVIDIA GTX 1080 GPU

layer size		vocabulary		batchsize	maxlen	GPU memory	speed	
embed	hidden	source	target			(peak)	sents/s	words/s
256	512	30000	30000	40	50	1.2 GB	174	4080
256	512	30000	60000	40	50	2.1 GB	148	3470
256	512	60000	60000	40	50	2.3 GB	145	3410
256	1024	60000	60000	40	50	2.7 GB	95	2220
512	1024	60000	60000	40	50	3.6 GB	79	1850
512	1024	60000	60000	80	50	4.9 GB	110	2600
512	1024	60000	60000	80	80	6.6 GB	87	2570

Hyperparameters: Efficiency

- hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- memory consumption affected by
  - number of model parameters
  - size of training instance (length · batchsize)
- we show some profile output for guidance:
  - Nematus ('test_train.sh')
  - NVIDIA GTX 1080 GPU

layer size		vocabulary		batchsize	maxlen	GPU memory	speed	
embed	hidden	source	target			(peak)	sents/s	words/s
256	512	30000	30000	40	50	1.2 GB	174	4080
256	512	30000	60000	40	50	2.1 GB	148	3470
256	512	60000	60000	40	50	2.3 GB	145	3410
256	1024	60000	60000	40	50	2.7 GB	95	2220
512	1024	60000	60000	40	50	3.6 GB	79	1850
512	1024	60000	60000	80	50	4.9 GB	110	2600
512	1024	60000	60000	80	80	6.6 GB	87	2570

Hyperparameters: Efficiency

- hyperparameters affect peak GPU memory and speed
- GPU memory is often the bottleneck in NMT training
- memory consumption affected by
  - number of model parameters
  - size of training instance (length · batchsize)
- we show some profile output for guidance:
  - Nematus ('test_train.sh')
  - NVIDIA GTX 1080 GPU

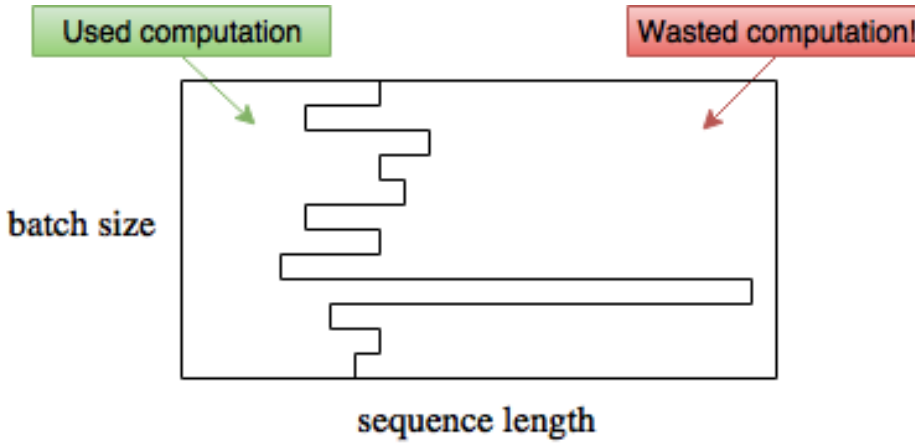
layer size		vocabulary		batchsize	maxlen	GPU memory	speed	
embed	hidden	source	target			(peak)	sents/s	words/s
256	512	30000	30000	40	50	1.2 GB	174	4080
256	512	30000	60000	40	50	2.1 GB	148	3470
256	512	60000	60000	40	50	2.3 GB	145	3410
256	1024	60000	60000	40	50	2.7 GB	95	2220
512	1024	60000	60000	40	50	3.6 GB	79	1850
512	1024	60000	60000	80	50	4.9 GB	110	2600
512	1024	60000	60000	80	80	6.6 GB	87	2570

Training: Minibatches

Why Minibatches?

- parallelization (GPUs!) is more efficiently with larger matrices
- easy way to increase matrix size: batch up training instances
- other advantage: stabilizes updates

- how do we deal with difference in sentence length in batch?
- standard solution: pad sentence with special tokens

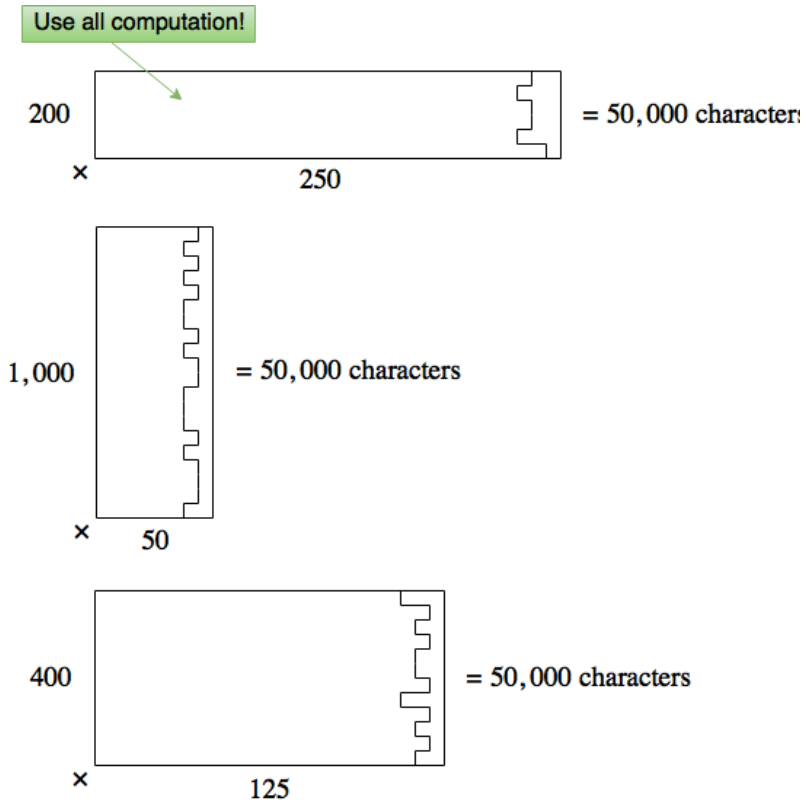


[Johansen et al., 2016]

Training: Minibatches

Speed-ups

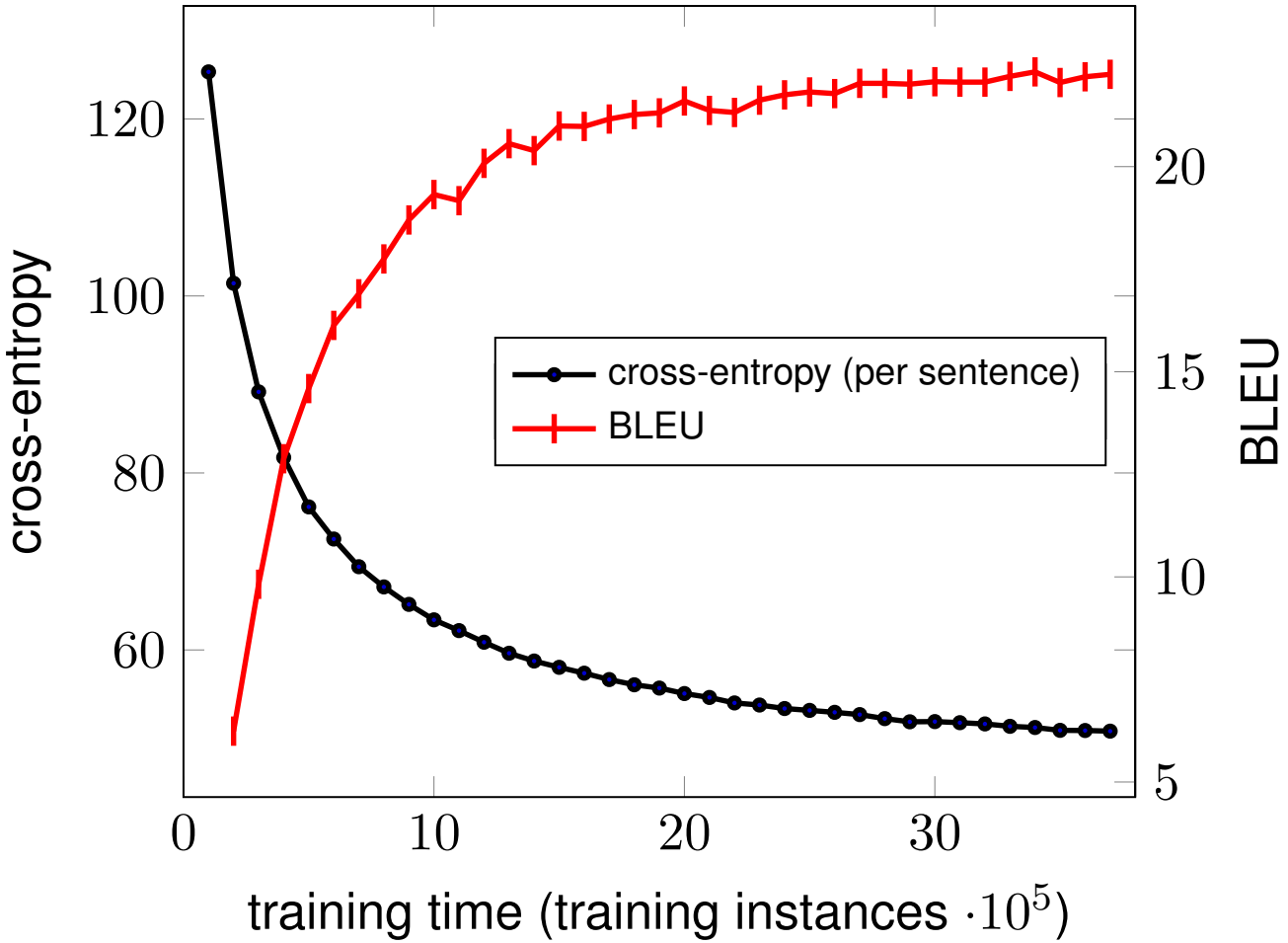
- sort sentences of same length together [Sutskever et al., 2014]
- adjust batch size depending on length [Johansen et al., 2016]



Out-of-memory: what to do

- little effect on quality:
  - reduce batch size
  - remove long sentences (also in validation!)
  - tie embedding layer and output layer in decoder [Press and Wolf, 2017] ('--tie_decoder_embeddings' in Nematus)
  - model parallelism: different parts of model on different GPU
- unknown (or negative) effect on quality:
  - reduce layer size
  - reduce target vocabulary

Training and Convergence



- BLEU more unstable than cross-entropy
- useful convergence criteria: BLEU early stopping

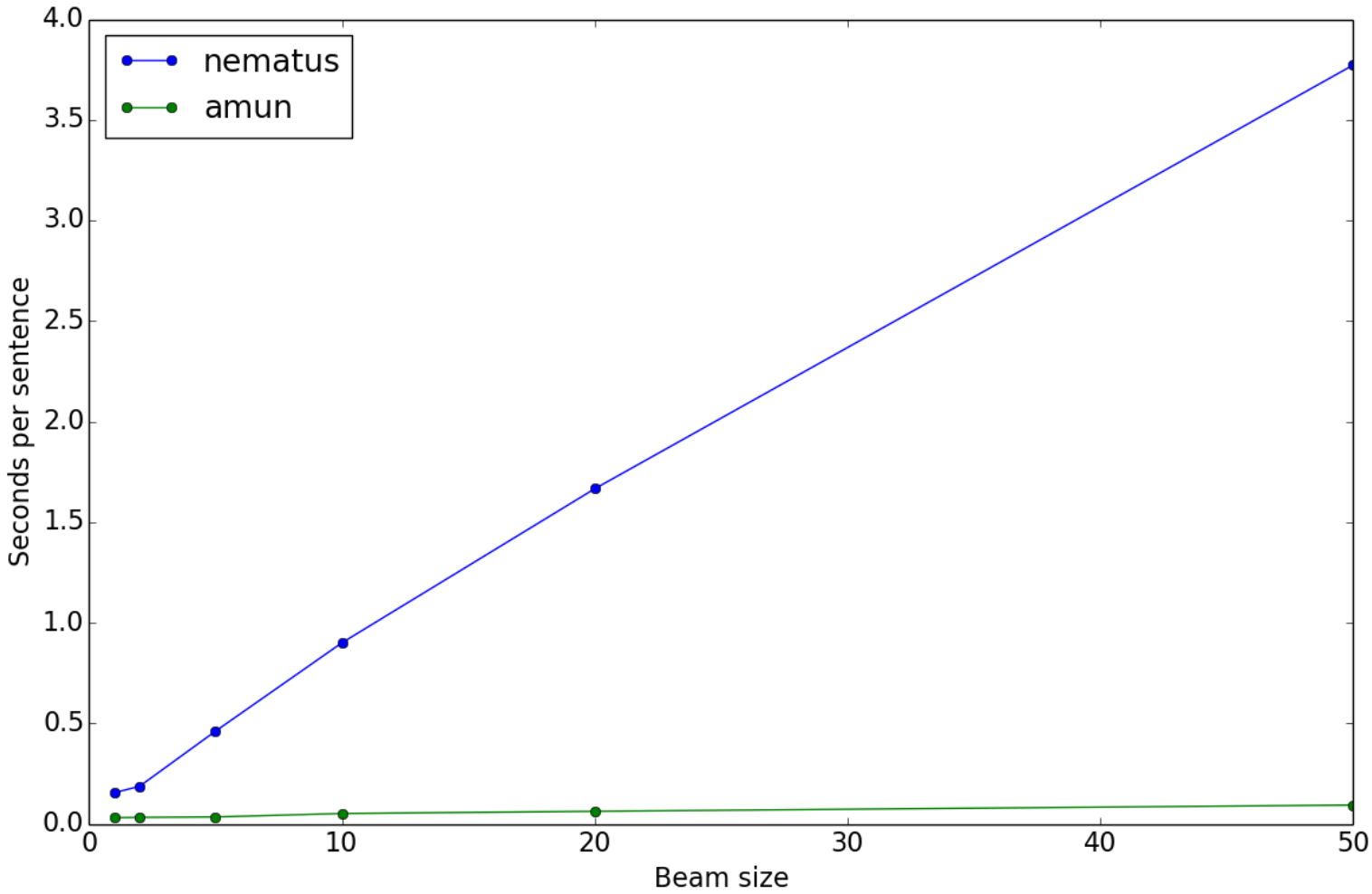
Decoding Efficiency

How to make decoding fast?

- Small beam size is often sufficient
- Greedy decoding can be competitive in quality
  - → especially with knowledge distillation [Kim and Rush, 2016]
- Filter output vocabulary [Jean et al., 2015, L'Hostis et al., 2016]
  based on which words commonly co-occur with source words
- process multiple sentences in batch [Wu et al., 2016]
- low-precision arithmetic [Wu et al., 2016]
  (requires suitable hardware)

NB: Amun supports batching, vocabulary filtering

Decoding Speed: Nematus vs. Amun



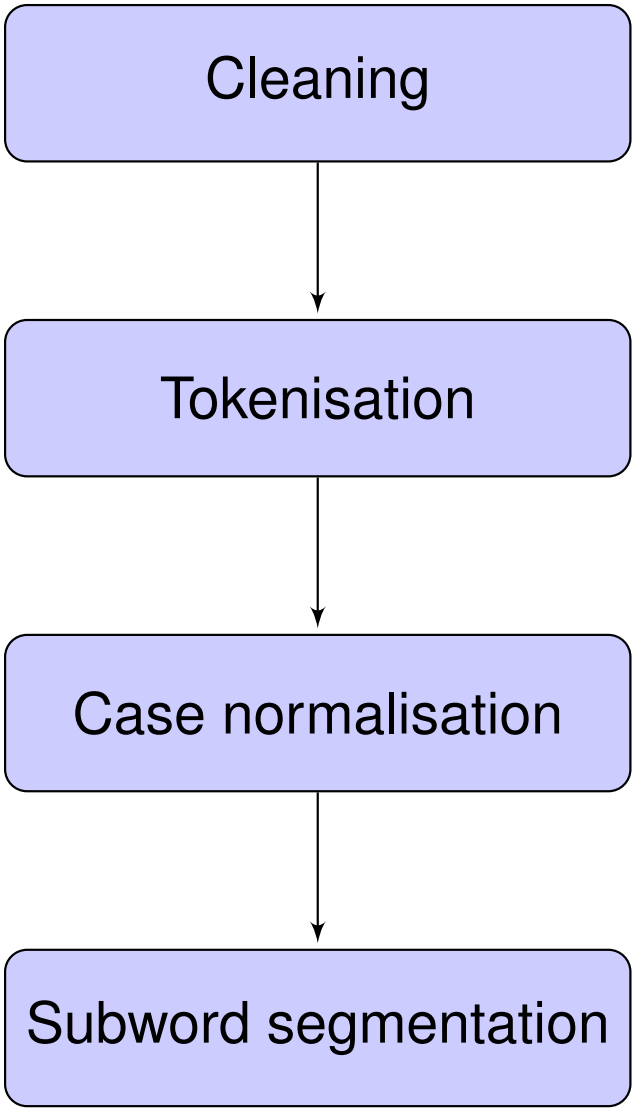
- Single GPU, single model, Titan X (Pascal)

Improving Translation Quality

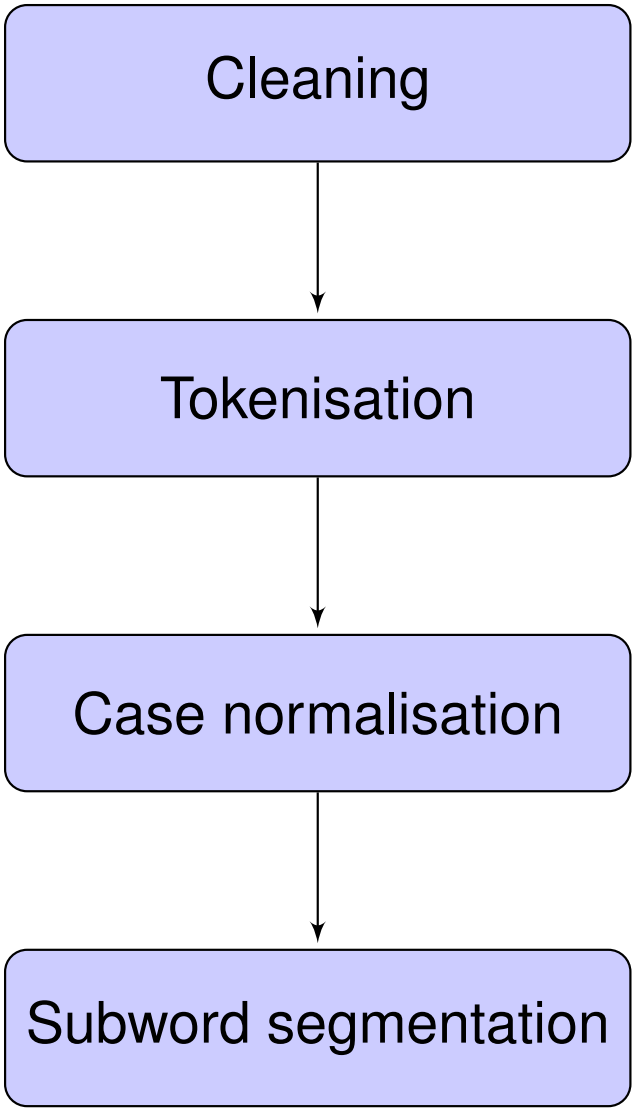
There are many possible ways of improving the basic system

1. Improve corpus preparation
2. Domain adaptation
3. Obtain appropriate synthetic data
4. Hybrid of NMT and traditional SMT
5. Add extra linguistic information
6. Minimum risk training
7. Deep models
8. Hyperparameter exploration

Corpus Preparation

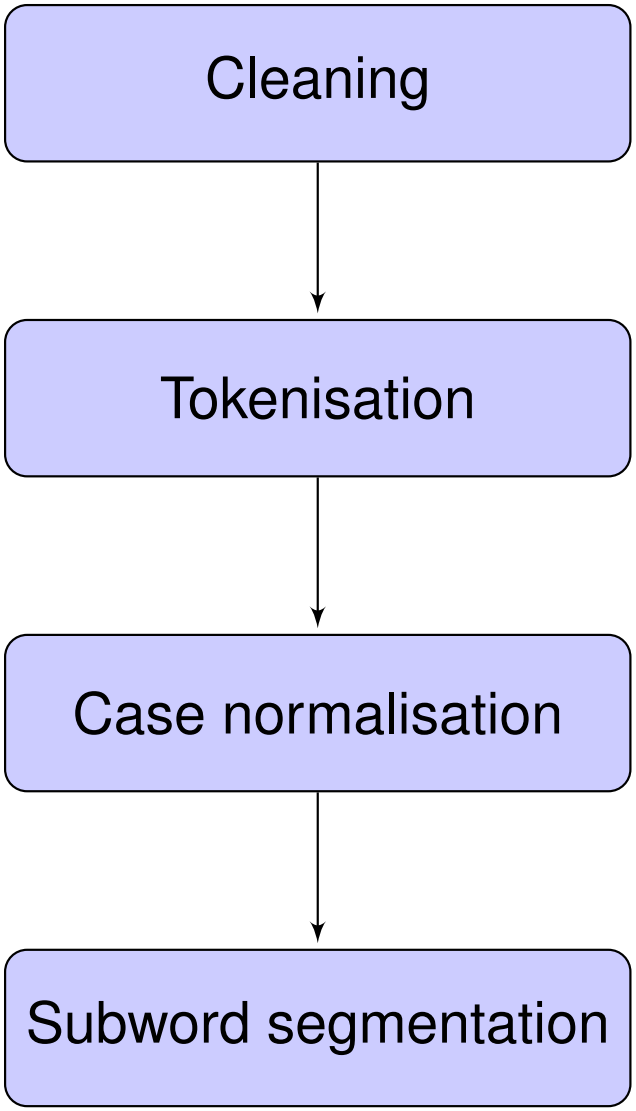


Corpus Preparation



- Punctuation/encoding/spelling normalisation
- Language identification
- Removing non-parallel segments

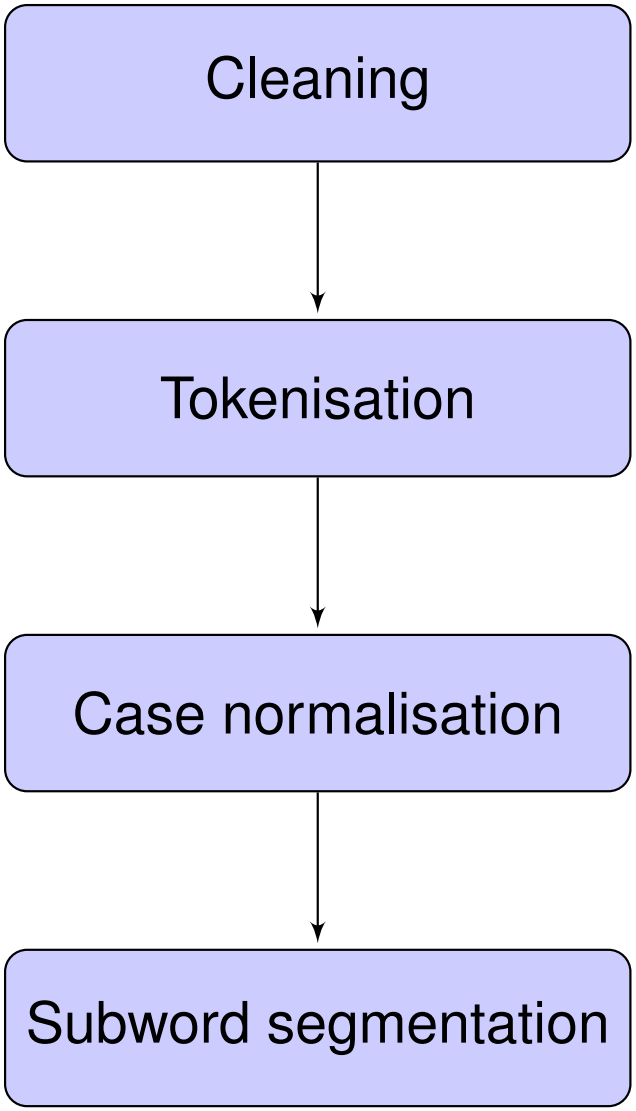
Corpus Preparation



- Punctuation/encoding/spelling normalisation
- Language identification
- Removing non-parallel segments

- Lowercasing
- Truecasing (convert to most frequent)
- Headlines etc.

Corpus Preparation



- Punctuation/encoding/spelling normalisation
- Language identification
- Removing non-parallel segments

- Lowercasing
- Truecasing (convert to most frequent)
- Headlines etc.

- Statistical
- Linguistically motivated

Effect of Noise in Training Data

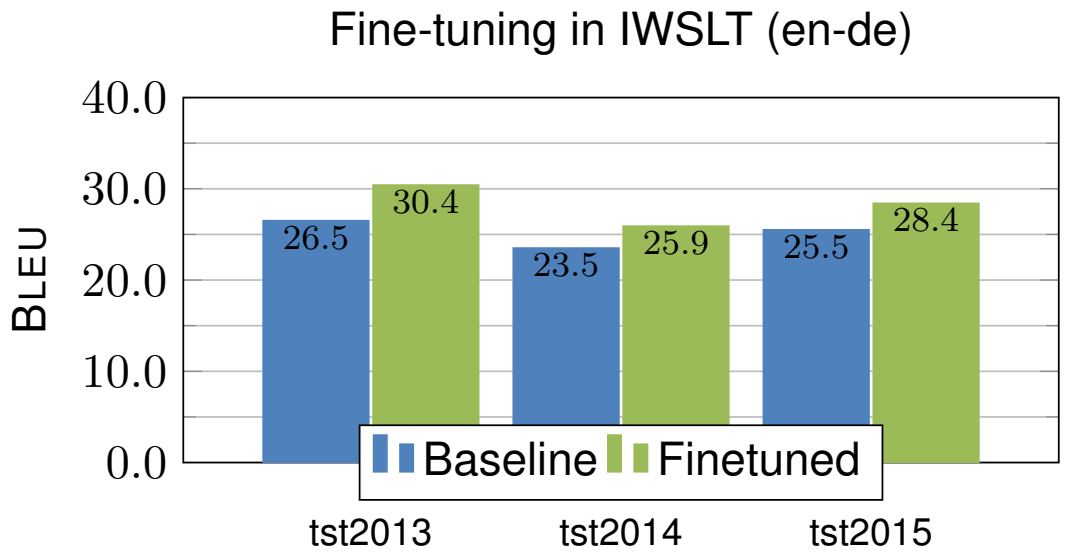
- [Chen et al., 2016] add noise to WMT EN-FR training data
- artificial noise: permute order of target sentences
- conclusion: NMT is more sensitive to (some types of) noise than SMT

	0%	10%	20%	50%
SMT	32.74	32.69	32.61	31.96
NMT	35.44	34.83	32.05	30.11

results from presentation of [Chen et al., 2016] at AMTA 2016

Domain adaptation with continued training

- SGD is sensitive to order of training instances
- best practice:
  - first train on all available data
  - continue training on in-domain data
- Large BLEU improvements reported with minutes of training time
  [Sennrich et al., 2016, Luong and Manning, 2015, Crego et al., 2016]



Generic system ($\approx$ 8M sentences), Fine-tune with TED ($\approx$ 200k)

Continued training with synthetic data

- what if we have monolingual in-domain training data?
- we compare fine-tuning with:
  - 200 000 sentence pairs in-domain
  - 200 000 target-language sentences in-domain, plus automatic back-translation

system	BLEU (tst2015)
WMT data	25.5
fine-tuned on in-domain (parallel)	28.4
fine-tuned on in-domain (synthetic)	26.7

English→German translation performance on IWSLT test set (TED talks).

→ parallel in-domain data is better, but domain adaptation with monolingual data is possible
→ WMT16 results (using large synthetic news corpora)

[Sennrich et al., 2016]

Continued training with synthetic data

Problem

How to create synthetic data from source-language in-domain data?

Continued training with synthetic data

Problem

How to create synthetic data from source-language in-domain data?

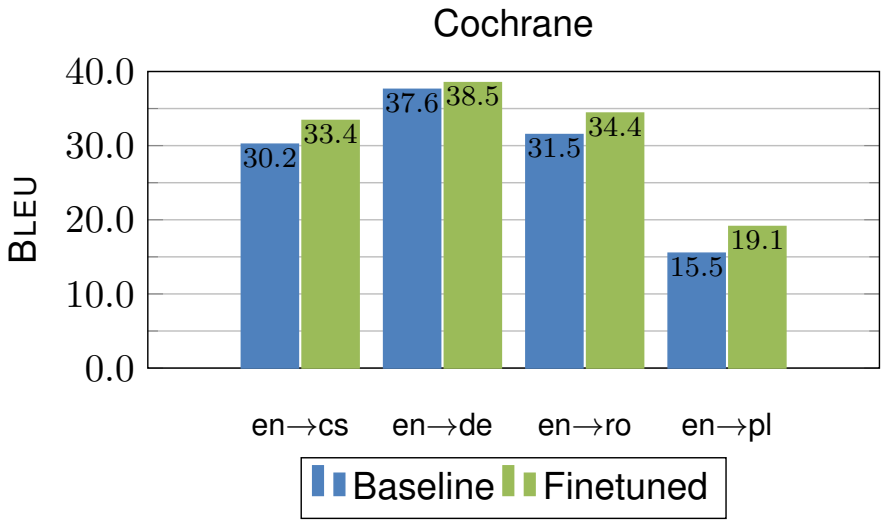
Solution

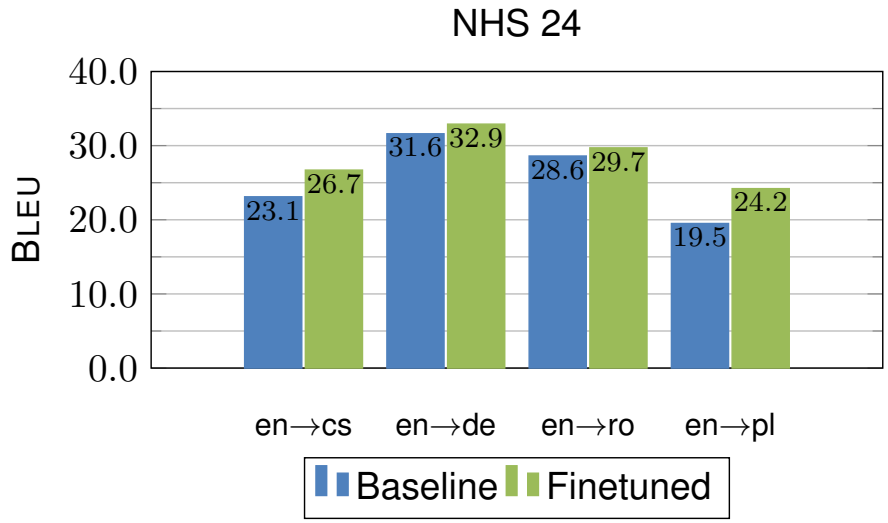
1. Gather source-language in-domain data.
2. Translate to target language
3. Use this translated data to select from CommonCrawl corpus
4. Back-translate selected data to create synthetic data

Continued training with synthetic data

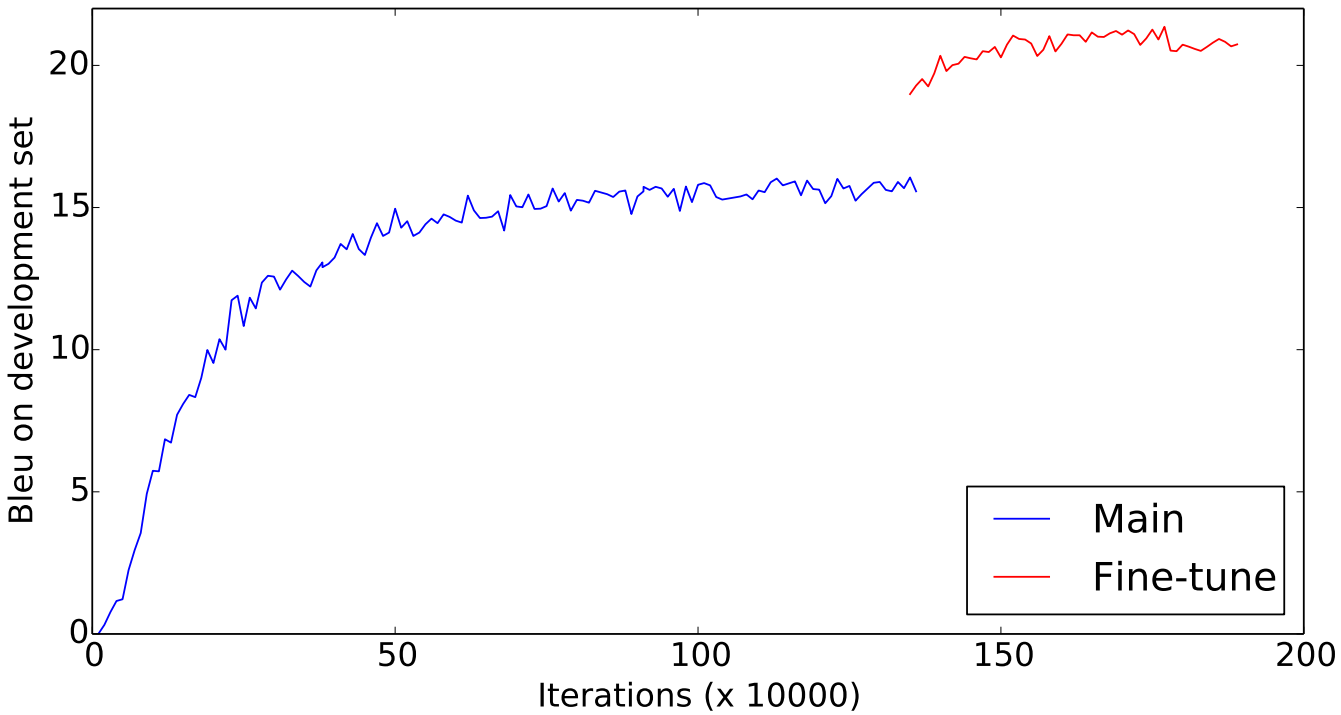
Setup

- Language pairs: English → Czech, German, Polish and Romanian
- Domains: Two healthcare websites (NHS 24 and Cochrane)
- Baselines: Data drawn from WMT releases and OPUS
- Fine-tuning:
  - Use crawls of full websites as selection “seed”
  - Continue training with 50-50 synthetic/parallel mix



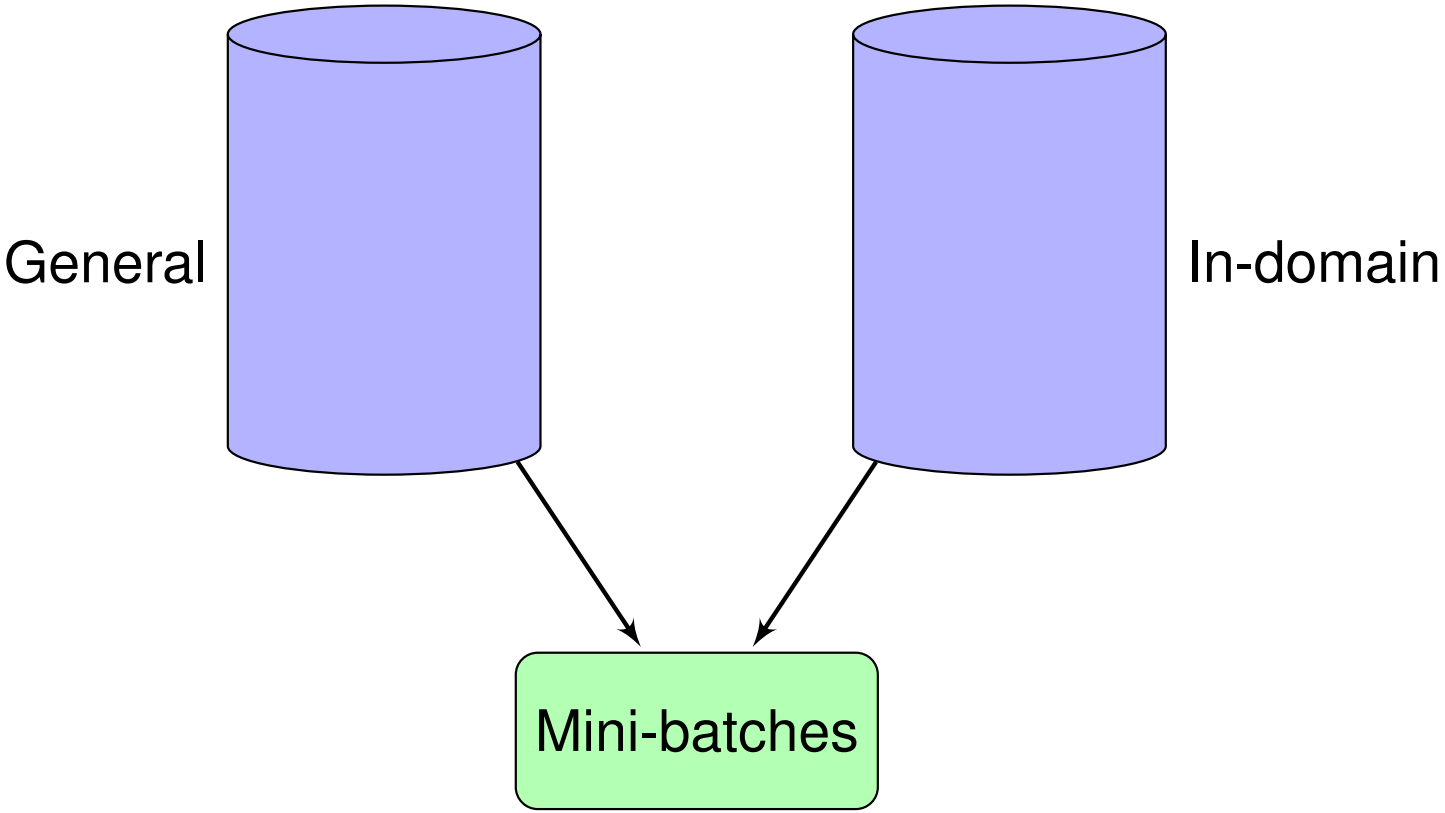


Continued Training with Synthetic Data: Sample Learning Curve



- English→Polish, select using Cochrane
- Main training on general domain, finetune on 50-50 mix

Nematus Domain Interpolation



- Use domain interpolation to mix general and in-domain

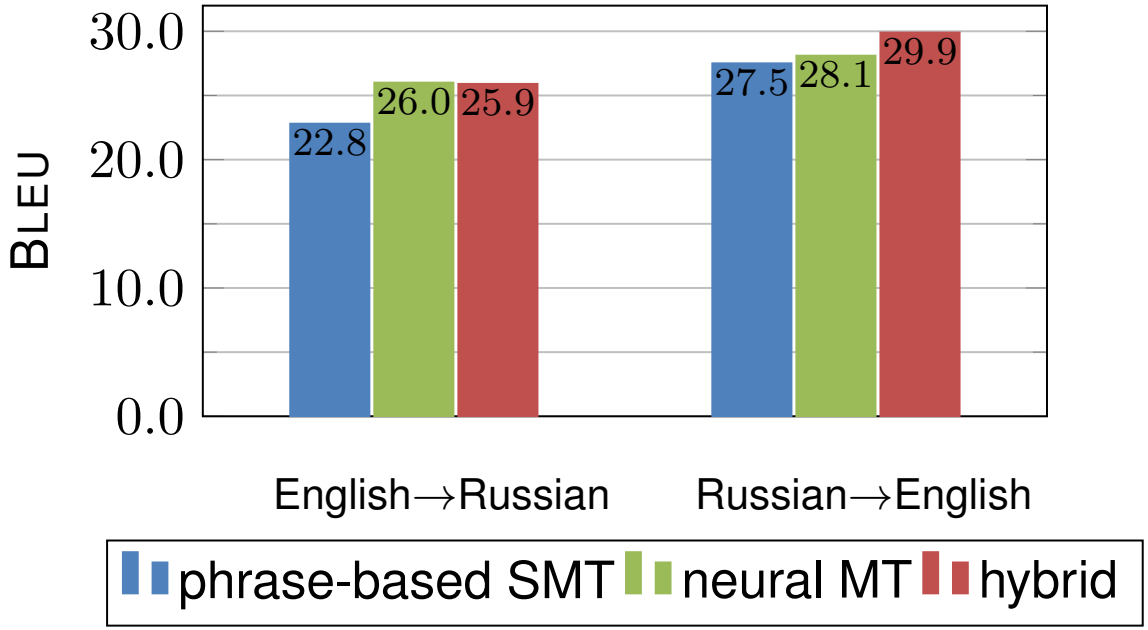
```
--use_domain_interpolation
--domain_interpolation_indomain_datasets
--domain_interpolation_(min|max)
--domain_interpolation_inc
```

NMT Hybrid Models

- Model combination (ensembling) is well established
- Several ways to combine NMT with PBMT / Syntax-based MT:
  - Re-ranking output of traditional SMT with NMT [Neubig et al., 2015]
  - Incorporating NMT as feature function in PBMT [Junczys-Dowmunt et al., 2016b]
  - Rescoring hiero lattices with NMT [Stahlberg et al., 2016]
- Reduces chance of “bizarre” NMT outputs

NMT Hybrid Models: Case Study

- NMT as feature function in PBMT [Junczys-Dowmunt et al., 2016b]
  → results depend on relative performance of PBMT and NMT



Why Linguistic Features?

disambiguate words by POS

English	German
$\text{close}_{\text{verb}}$	schließen
$\text{close}_{\text{adj}}$	nah
$\text{close}_{\text{noun}}$	Ende

source	*We thought a win like this might be* $\text{close}_{\text{adj}}$*.*
reference	*Wir dachten, dass ein solcher Sieg nah sein könnte.*
baseline NMT	**Wir dachten, ein Sieg wie dieser könnte schließen.*

Why Linguistic Features?

better generalization; combat data sparsity

word form
liegen (lie)
liegst (lie)
lag (lay)
läge (lay)

Why Linguistic Features?

better generalization; combat data sparsity

word form	lemma	morph. features
liegen (lie)	liegen (lie)	(3.p.pl. present)
liegst (lie)	liegen (lie)	(2.p.sg. present)
lag (lay)	liegen (lie)	(3.p.sg. past)
läge (lay)	liegen (lie)	(3.p.sg. subjunctive II)

Neural Machine Translation: Multiple Input Features

Use separate embeddings for each feature, then concatenate

baseline: only word feature

$$E(close) = \begin{bmatrix} 0.5 \\ 0.2 \\ 0.3 \\ 0.1 \end{bmatrix}$$

$|F|$ input features

$$E_1(close) = \begin{bmatrix} 0.4 \\ 0.1 \\ 0.2 \end{bmatrix} \qquad E_2(adj) = \begin{bmatrix} 0.1 \end{bmatrix} \qquad E_1(close) \parallel E_2(adj) = \begin{bmatrix} 0.4 \\ 0.1 \\ 0.2 \\ 0.1 \end{bmatrix}$$

Experiments

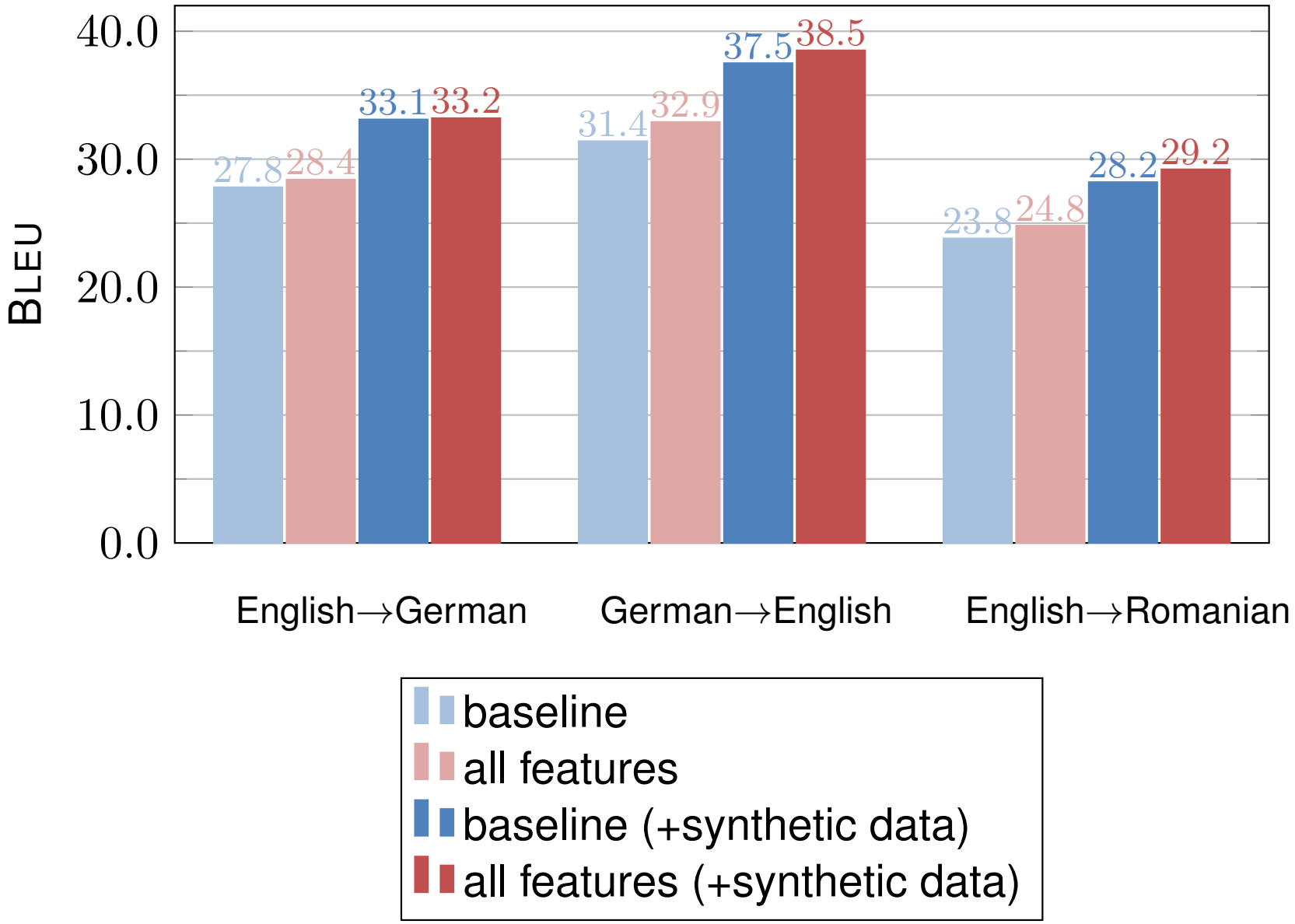
Features

- lemmas
- morphological features
- POS tags
- dependency labels
- BPE tags

Data

- WMT16 training/test data
- English$\leftrightarrow$German and English$\rightarrow$Romanian

Results: BLEU ↑



Minimum Risk Training [Shen et al., 2016]

- The standard NMT training objective is cross-entropy
  - $\rightarrow$ maximise probability of training data
- In traditional SMT, we usually tune for BLEU
- Can train NMT to minimise Expected Loss

$$\sum_{s=1}^{S} \mathbb{E}_{p(y|x^{(s)})} \left[\Delta(y, y^{(s)}) \right]$$

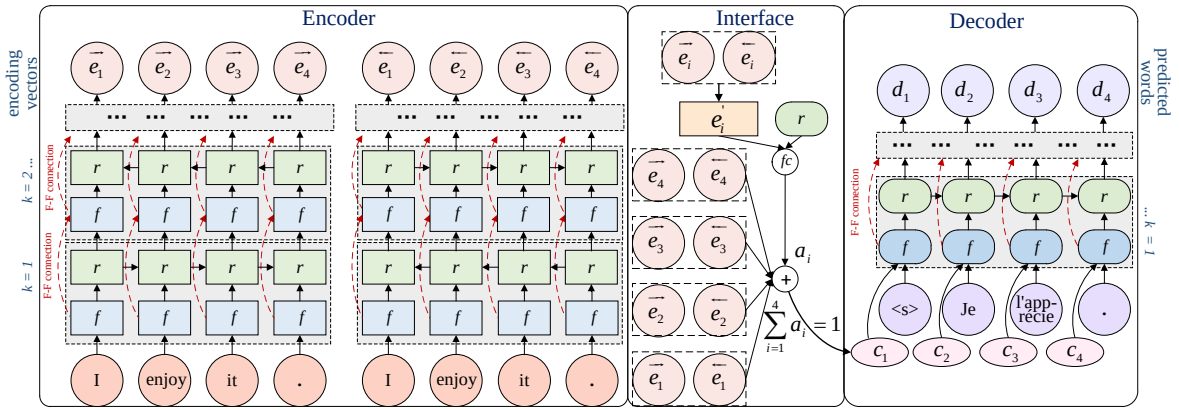
(Loss function: Δ ; Training pair: $(x^{(s)}, y^{(s)})$)

- Run MRT after training with cross-entropy loss
- Approximate expectation with sum over samples

Minimum Risk Training in Nematus

- Recipe:
  - Train initial model with standard cross-entropy training
  - Continue training with '--objective MRT'
- Sensitive to hyperparameters
  $\rightarrow$ Use small learning rate with SGD
- mixed results:
  - Improvements over some baselines (EN$\rightarrow$RO parallel)
  - No improvement so far over others (EN$\rightarrow$DE with synthetic data)

Deep Models



deep architecture by [Zhou et al., 2016]

- deep recurrent architectures [Zhou et al., 2016, Wu et al., 2016]
- [Zhou et al., 2016] report +4 BLEU from 16 RNN layers:
  (9 encoder; 7 decoder)
- important trick: residual connections
- challenges: efficiency; memory limitations

Hyperparameter Exploration

Massive Exploration of Neural Machine Translation Architectures [Britz et al., 2017]

- Spent 250,000 hours GPU time exploring hyperparameters
- Conclusions:
  - Small gain from increasing embedding size
  - LSTM better than GRU
  - 2-4 Layer Bidirectional encoder better
  - 4-layer decoder gives some advantage
  - Additive better than multiplicative attention
  - Large beams not helpful (best = 10)
- BLEU variance across runs small ($\pm$ 0.2-0.3)

Practical Neural Machine Translation

Getting Started: Do it Yourself

- sample files and instructions for training NMT model
  `https://github.com/rsennrich/wmt16-scripts`
- pre-trained models to test decoding (and for further experiments)
  `http://statmt.org/rsennrich/wmt16_systems/`
- lab on installing/using Nematus:
  `http://www.statmt.org/eacl2017/practical-nmt-lab.pdf`

(A small selection of) Resources

NMT tools

- Nematus (theano) https://github.com/rsennrich/nematus
- OpenNMT (torch) https://github.com/OpenNMT/OpenNMT
- nmt.matlab https://github.com/lmthang/nmt.matlab
- neural monkey (tensorflow) https://github.com/ufal/neuralmonkey
- lamtram (DyNet) https://github.com/neubig/lamtram
- ...and many more https://github.com/jonsafari/nmt-list

Further Reading

secondary literature

- lecture notes by Kyunghyun Cho: [Cho, 2015]
- chapter on *Neural Network Models* in "Statistical Machine Translation" by Philipp Koehn `http://mt-class.org/jhu/assets/papers/neural-network-models.pdf`
- tutorial on sequence-to-sequence models by Graham Neubig

  `https://arxiv.org/abs/1703.01619`

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements 645452 (QT21) and 644402 (HimL).

Questions

Thank you!

Bibliography I

Allen, R. (1987).

Several Studies on Natural Language and Back-Propagation.
In IEEE First International Conference on Neural Networks, pages 335–341, San Diego, California, USA.

Bahdanau, D., Cho, K., and Bengio, Y. (2015).

Neural Machine Translation by Jointly Learning to Align and Translate.
In Proceedings of the International Conference on Learning Representations (ICLR).

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003).

A Neural Probabilistic Language Model.
J. Mach. Learn. Res., 3:1137–1155.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2016).

Neural versus Phrase-Based Machine Translation Quality: a Case Study.
In EMNLP 2016.

Britz, D., Goldie, A., Luong, T., and Le, Q. (2017).

Massive Exploration of Neural Machine Translation Architectures.
ArXiv e-prints.

Chen, B., Kuhn, R., Foster, G., Cherry, C., and Huang, F. (2016).

Bilingual Methods for Adaptive Training Data Selection for Machine Translation.
In Proceedings of AMTA.

Cho, K. (2015).

Natural Language Understanding with Distributed Representation.
CoRR, abs/1511.07916.

Bibliography II


Cho, K., Courville, A., and Bengio, Y. (2015).
Describing Multimedia Content using Attention-based Encoder-Decoder Networks.


Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014).
On the Properties of Neural Machine Translation: Encoder-Decoder Approaches.
ArXiv e-prints.


Crego, J., Kim, J., Klein, G., Rebollo, A., Yang, K., Senellart, J., Akhanov, E., Brunelle, P., Coquard, A., Deng, Y., Enoue, S., Geiss, C., Johanson, J., Khalsa, A., Khiari, R., Ko, B., Kobus, C., Lorieux, J., Martins, L., Nguyen, D.-C., Priori, A., Riccardi, T., Segal, N., Servan, C., Tiquet, C., Wang, B., Yang, J., Zhang, D., Zhou, J., and Zoldan, P. (2016).
SYSTRAN's Pure Neural Machine Translation Systems.
ArXiv e-prints.


Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. (2014).
Fast and Robust Neural Network Joint Models for Statistical Machine Translation.
In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1370–1380, Baltimore, Maryland. Association for Computational Linguistics.


Gage, P. (1994).
A New Algorithm for Data Compression.
C Users J., 12(2):23–38.


Gal, Y. (2015).
A Theoretically Grounded Application of Dropout in Recurrent Neural Networks.
ArXiv e-prints.


Ha, T.-L., Niehues, J., Cho, E., Mediani, M., and Waibel, A. (2015).
The KIT translation systems for IWSLT 2015.
In Proceedings of the International Workshop on Spoken Language Translation (IWSLT), pages 62–69.

Bibliography III

Haddow, B., Huck, M., Birch, A., Bogoychev, N., and Koehn, P. (2015).

The Edinburgh/JHU Phrase-based Machine Translation Systems for WMT 2015.
In Proceedings of the Tenth Workshop on Statistical Machine Translation, pages 126–133, Lisbon, Portugal. Association for Computational Linguistics.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015).

On Using Very Large Target Vocabulary for Neural Machine Translation.
In
Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference o
pages 1–10, Beijing, China. Association for Computational Linguistics.

Johansen, A. R., Hansen, J. M., Obeid, E. K., Sønderby, C. K., and Winther, O. (2016).

Neural Machine Translation with Characters and Hierarchical Encoding.
CoRR, abs/1610.06550.

Junczys-Dowmunt, M., Dwojak, T., and Hoang, H. (2016a).

Is Neural Machine Translation Ready for Deployment? A Case Study on 30 Translation Directions.
In Proceedings of IWSLT.

Junczys-Dowmunt, M., Dwojak, T., and Sennrich, R. (2016b).

The AMU-UEDIN Submission to the WMT16 News Translation Task: Attention-based NMT Models as Feature Functions in Phrase-based SMT.
In Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 316–322, Berlin, Germany. Association for Computational Linguistics.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2016).

Log-linear Combinations of Monolingual and Bilingual Neural Machine Translation Models for Automatic Post-Editing.
In Proceedings of the First Conference on Machine Translation, pages 751–758, Berlin, Germany. Association for Computational Linguistics.

Bibliography IV

Kalchbrenner, N. and Blunsom, P. (2013).

Recurrent Continuous Translation Models.
In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle. Association for Computational Linguistics.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014).

A Convolutional Neural Network for Modelling Sentences.
In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).

Kim, Y. and Rush, A. M. (2016).

Sequence-Level Knowledge Distillation.
CoRR, abs/1606.07947.

Lee, J., Cho, K., and Hofmann, T. (2016).

Fully Character-Level Neural Machine Translation without Explicit Segmentation.
ArXiv e-prints.

L'Hostis, G., Grangier, D., and Auli, M. (2016).

Vocabulary Selection Strategies for Neural Machine Translation.
ArXiv e-prints.

Luong, M.-T. and Manning, C. D. (2015).

Stanford Neural Machine Translation Systems for Spoken Language Domains.
In Proceedings of the International Workshop on Spoken Language Translation 2015, Da Nang, Vietnam.

Luong, T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014).

Addressing the Rare Word Problem in Neural Machine Translation.
CoRR, abs/1410.8206.

Bibliography V


Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010).
Recurrent neural network based language model.
In
INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, Se
pages 1045–1048.


Neubig, G., Morishita, M., and Nakamura, S. (2015).
Neural Reranking Improves Subjective Quality of Machine Translation: NAIST at WAT2015.
In Proceedings of the 2nd Workshop on Asian Translation (WAT2015), pages 35–41, Kyoto, Japan.


Press, O. and Wolf, L. (2017).
Using the Output Embedding to Improve Language Models.
In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL),
Valencia, Spain.


Schwenk, H., Dechelotte, D., and Gauvain, J.-L. (2006).
Continuous Space Language Models for Statistical Machine Translation.
In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 723–730, Sydney, Australia.


Sennrich, R. (2017).
How Grammatical is Character-level Neural Machine Translation? Assessing MT Quality with Contrastive Translation Pairs.
In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL),
Valencia, Spain.


Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., Läubli, S., Barone, A. V. M., and
Nădejde, M. (2017).
Nematus: a Toolkit for Neural Machine Translation.
In Proceedings of EACL (Demo Session).

Bibliography VI

Sennrich, R. and Haddow, B. (2015).

A Joint Dependency Model of Morphological and Syntactic Structure for Statistical Machine Translation.
In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2081–2087, Lisbon, Portugal. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016).

Improving Neural Machine Translation Models with Monolingual Data.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016).

Minimum Risk Training for Neural Machine Translation.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).

Stahlberg, F., Hasler, E., Waite, A., and Byrne, B. (2016).

Syntactically Guided Neural Machine Translation.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 299–305, Berlin, Germany. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014).

Sequence to Sequence Learning with Neural Networks.
In
Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, pages 3104–3112, Montreal, Quebec, Canada.

Vaswani, A., Zhao, Y., Fossum, V., and Chiang, D. (2013).

Decoding with Large-Scale Neural Language Models Improves Translation.
In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, pages 1387–1392, Seattle, Washington, USA.

Bibliography VII

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016).
Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.
ArXiv e-prints.

Zhou, J., Cao, Y., Wang, X., Li, P., and Xu, W. (2016).
Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation.
Transactions of the Association of Computational Linguistics – Volume 4, Issue 1, pages 371–383.