THE UNIVERSITY *of* EDINBURGH

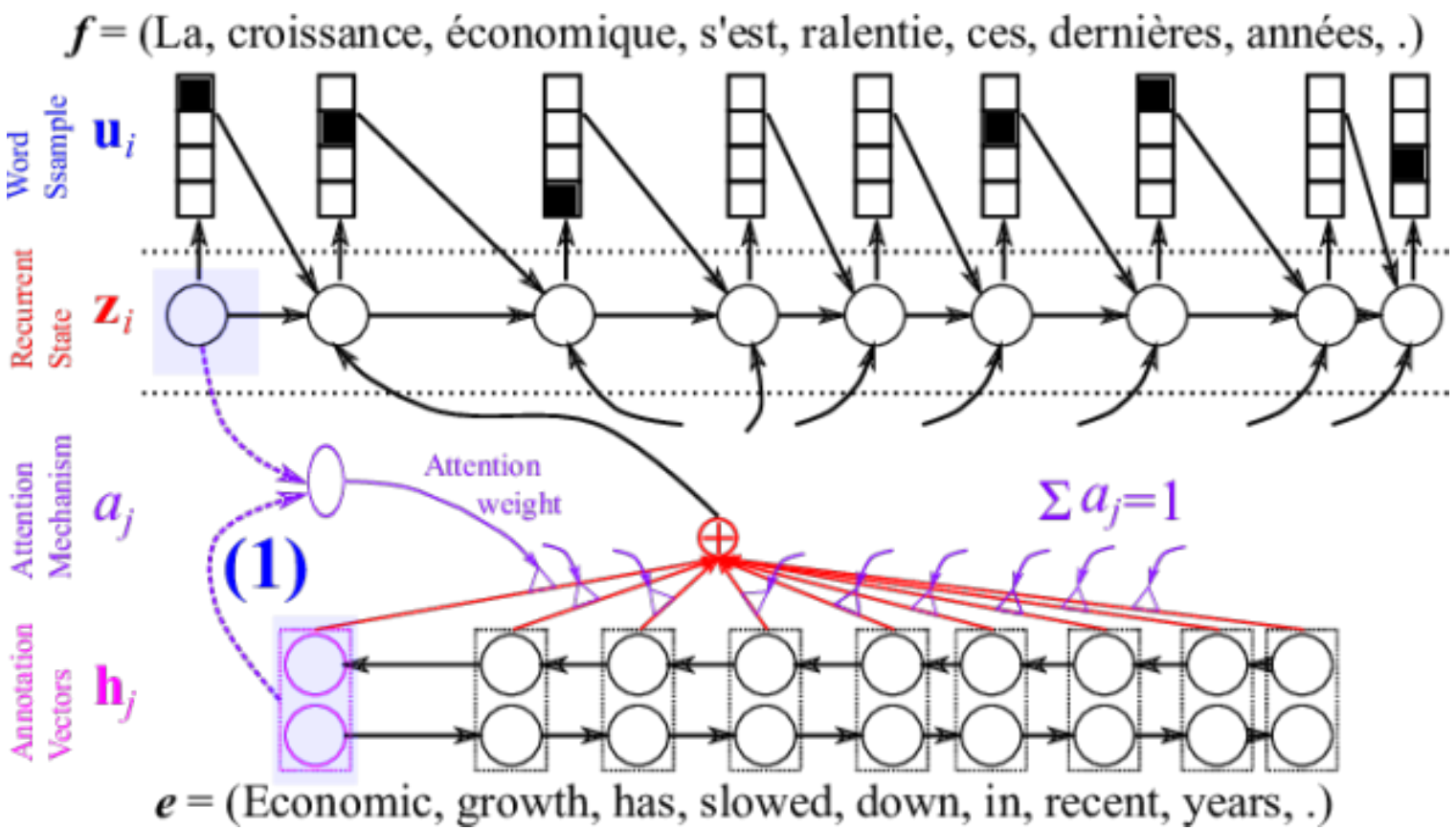
Neural Machine Translation

Rico Sennrich

Institute for Language, Cognition and Computation
University of Edinburgh

May 18 2016

Neural Machine Translation



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

1. neural network crash course

2. introduction to neural machine translation
   - neural language models
   - attentional encoder-decoder

3. recent research, opportunities and challenges in neural machine translation

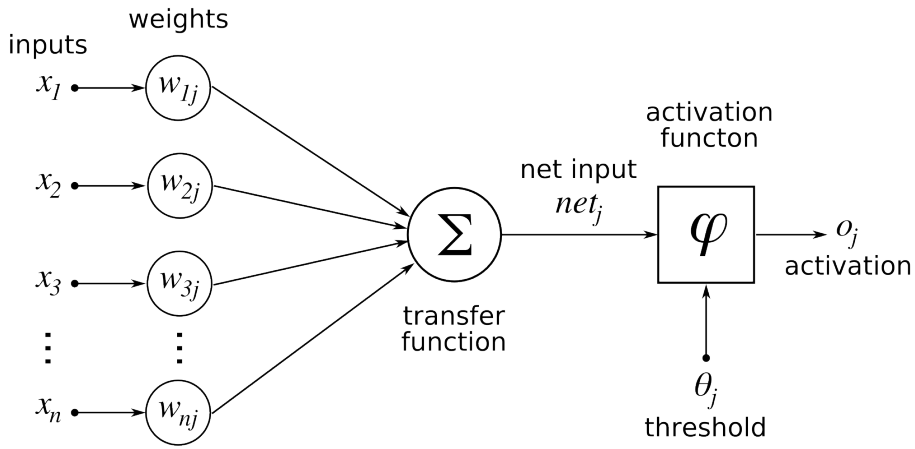
Neural Machine Translation

1. neural network crash course

2. introduction to neural machine translation
   - neural language models
   - attentional encoder-decoder

3. recent research, opportunities and challenges in neural machine translation

Building block: artificial neurons



analogy to biological neurons

- input $\approx$ dendrites
- activation function $\approx$ neuron 'fires' if voltage threshold is reached
- output $\approx$ axon

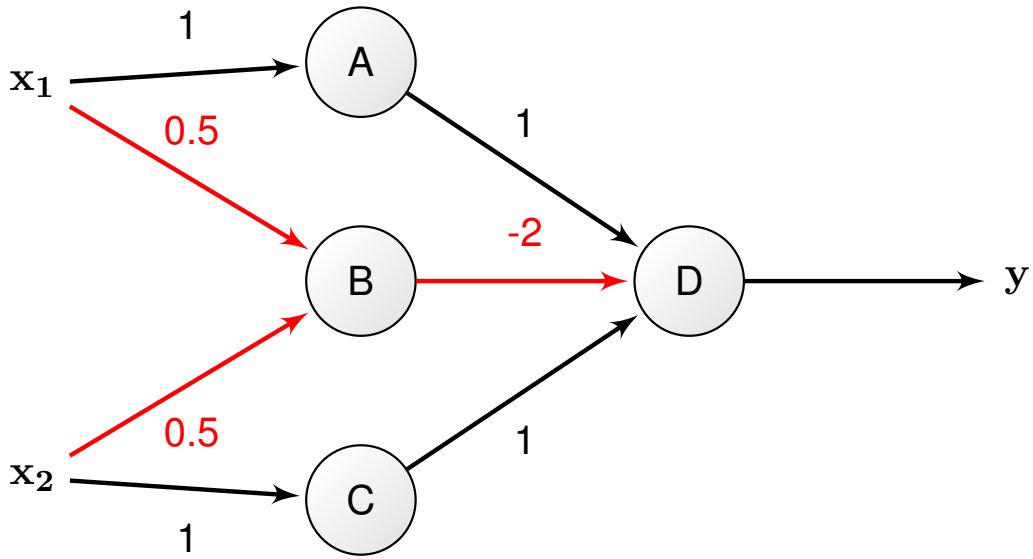
A simple neural network

neural networks can solve non-linear functions.

XOR

Truth table

A	B	output
0	0	0
0	1	1
1	0	1
1	1	0

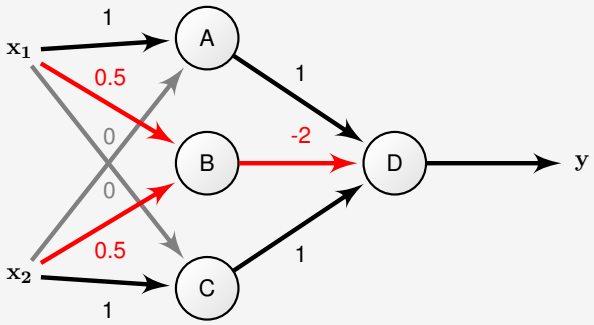


(neurons arranged in layers, and fire if input is ≥ 1)

A simple neural network: math

neural networks can be implemented via matrix operations

network



$$w_1 = \begin{bmatrix} 1 & 0 \\ 0.5 & 0.5 \\ 0 & 1 \end{bmatrix} \quad h_1 = \begin{bmatrix} A \\ B \\ C \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$w_2 = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \quad y = \begin{bmatrix} D \end{bmatrix}$$

calculation of $x \mapsto y$

$$\varphi(z) = z \geq 1$$

$$h_1 = \varphi(w_1 x)$$

$$y = \varphi(w_2 h_1)$$

A simple neural network: Python code

```
import numpy

#activation function
def phi(x):
    return numpy.greater_equal(x,1).astype(int)

def nn(x):
    w1 = numpy.array([ [1, 0.5, 0], [0, 0.5, 1] ])
    w2 = numpy.array([[1], [-2], [1]])
    h1 = phi(x.dot(w1))
    h2 = phi(h1.dot(w2))
    return h2

print nn(numpy.array([1, 0]))
```

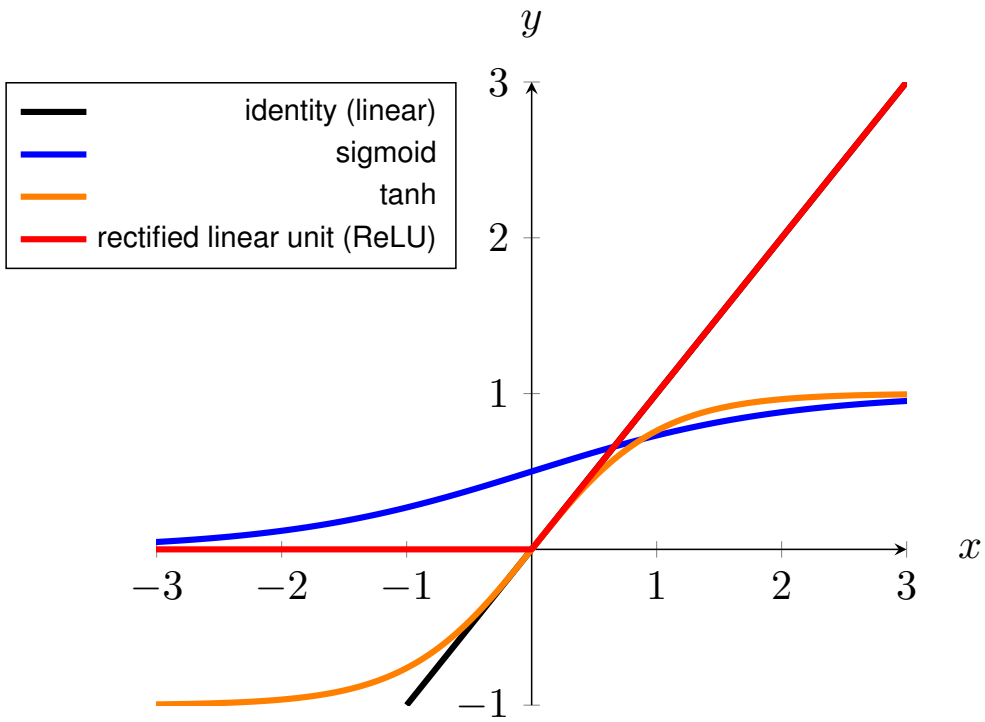
Training a neural network

Gradient descent

- requirements:
  - labelled training data (supervised learning)
  - differentiable objective function
- in forward pass, compute network output
- compare output to true label to compute error
- move all parameters in direction that minimizes error
- chain rule allows efficient computation of gradient for each parameter in backward pass $\rightarrow$ **backpropagation** of error
- we approximate gradient on small minibatches to perform frequent updates $\rightarrow$ **stochastic** gradient descent

Activation functions

- desirable:
  - differentiable (for stochastic gradient descent)
  - monotonic (for convexity)
  - non-linear activation functions essential to learn non-linear functions



Further basics

- hyperparameters:
  - number and size of layers
  - minibatch size
  - learning rate
  - ...
- initialisation of weight matrices
- stopping criterion
- regularization (dropout)
- bias units (always-on input)
- more complex architectures (recurrent/convolutional networks)

Resources

- Theano `http://deeplearning.net/software/theano/`
- Torch `http://torch.ch/`
- Tensorflow `https://www.tensorflow.org/`
- toolkits provide useful abstractions for neural networks:
  - routines for n-dimensional arrays (tensors)
  - simple use of different linear algebra backends (CPU/GPU)
  - automatic differentiation

Neural Machine Translation

1. neural network crash course

2. introduction to neural machine translation
   - neural language models
   - attentional encoder-decoder

3. recent research, opportunities and challenges in neural machine translation

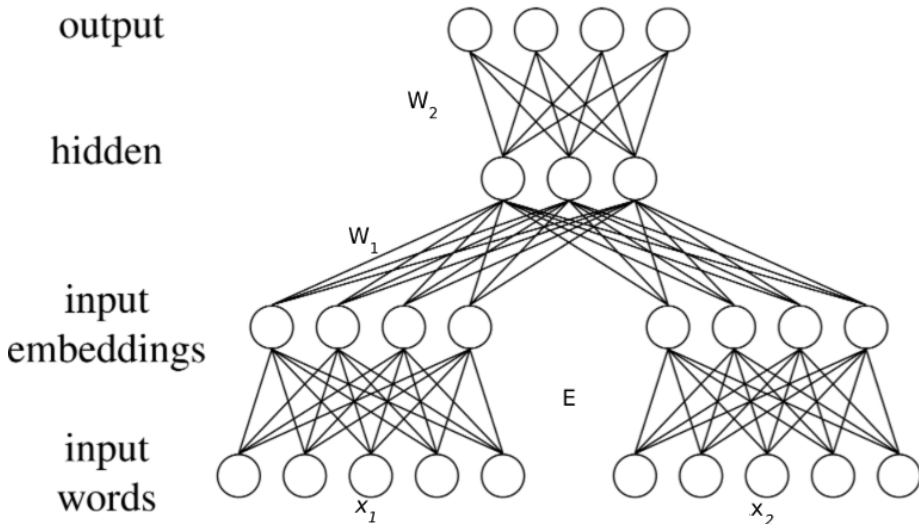
Language modelling

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

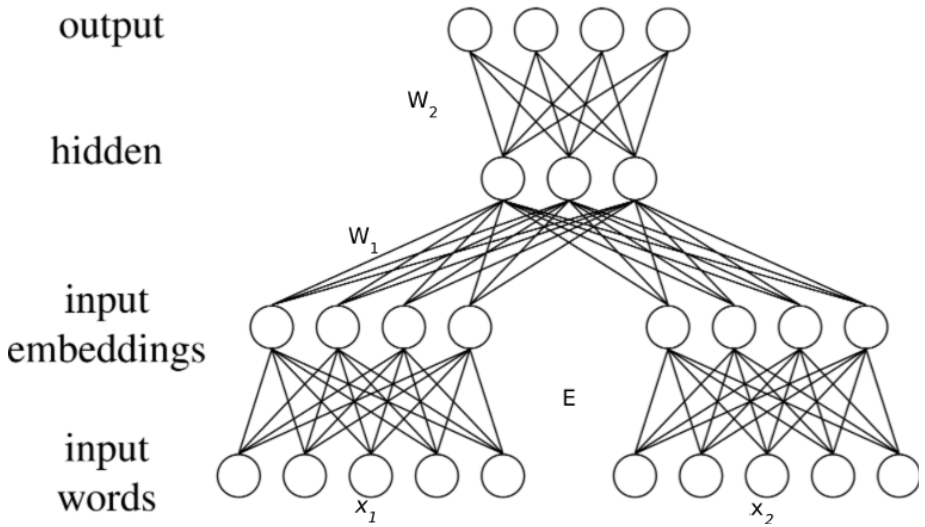
Softmax activation function

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
- scale well to many input words
  $\rightarrow$ models with source context [Devlin et al., 2014]

Recurrent neural network language model (RNNLM)

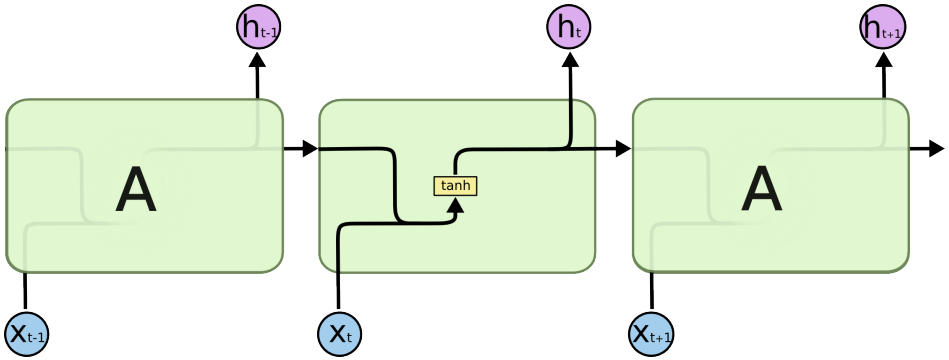
RNNLM

- 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

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

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

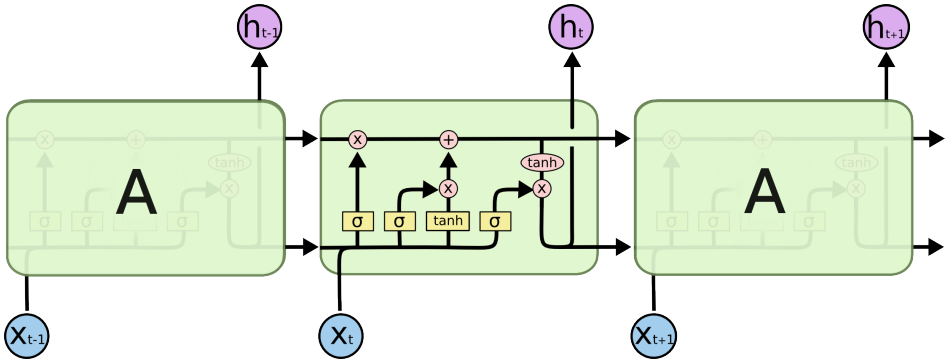
RNN variants



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)

Neural Machine Translation

1. neural network crash course
2. introduction to neural machine translation
   - neural language models
   - attentional encoder-decoder
3. recent research, opportunities and challenges in neural machine translation

Translation modelling

decomposition of translation problem (for NMT)

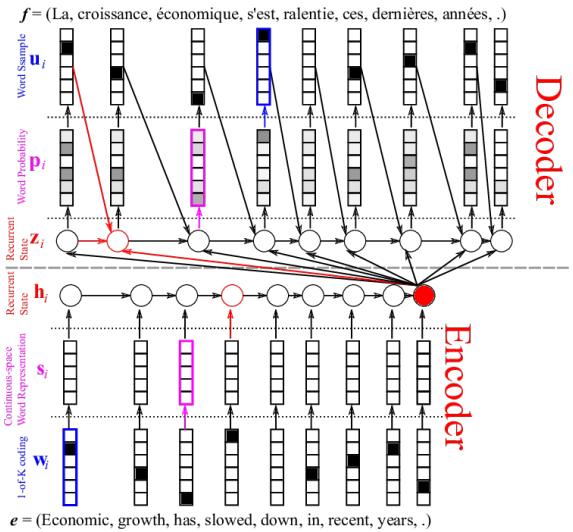
- a source sentence S of length m is a sequence $x_1, \ldots, x_m$
- a target sentence T of length n is a sequence $y_1, \ldots, y_n$

$$
\begin{aligned}
T^* &= \arg\max_t p(T|S) \\
p(T|S) &= p(y_1, \ldots, y_n | x_1, \ldots, x_m) \\
&= \prod_{i=1}^{n} p(y_i | y_0, \ldots, y_{i-1}, x_1, \ldots, x_m)
\end{aligned}
$$

Translating with RNNs

Encoder-decoder [Sutskever et al., 2014, Cho et al., 2014]

- two RNNs (LSTM or GRU):
  - **encoder** reads input and produces hidden state representations
  - **decoder** produces output, based on last encoder hidden state
- encoder and decoder are learned jointly
  $\rightarrow$ supervision signal from parallel text is backpropagated

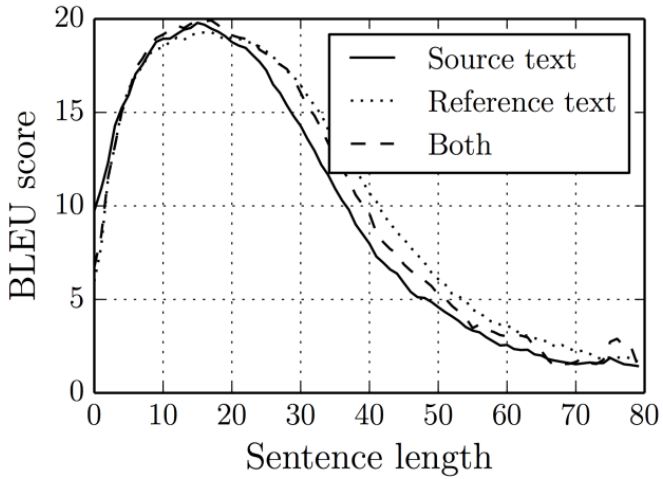


Kyunghyun Cho http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-2/

Neural machine translation: information bottleneck

summary vector

- last encoder hidden-state “summarizes” source sentence
- can fixed-size vector represent meaning of arbitrarily long sentence?
- empirically, quality decreases for long sentences
- reversing source sentence brings some improvement [Sutskever et al., 2014]

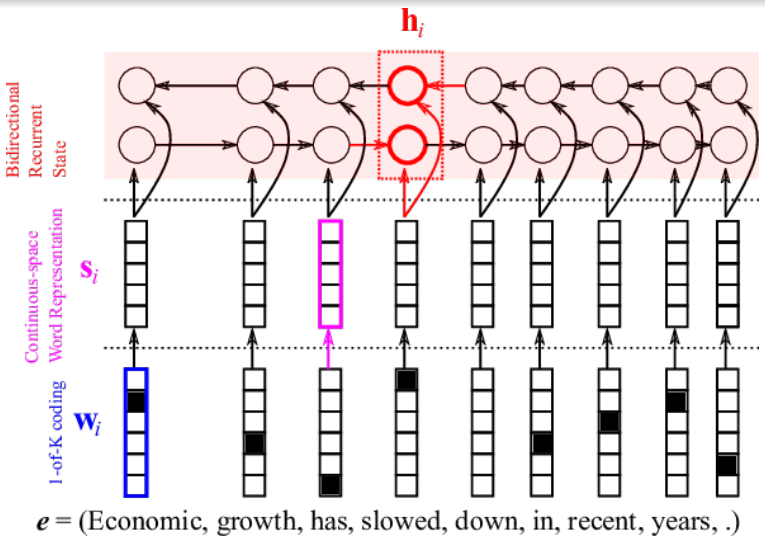


[Sutskever et al., 2014]

Attentional encoder-decoder

encoder

- goal: avoid bottleneck of summary vector
- use bidirectional RNN, and concatenate forward and backward states
  $\rightarrow$ *annotation vector* h_i
- represent source sentence as vector of n annotations
  $\rightarrow$ variable-length representation

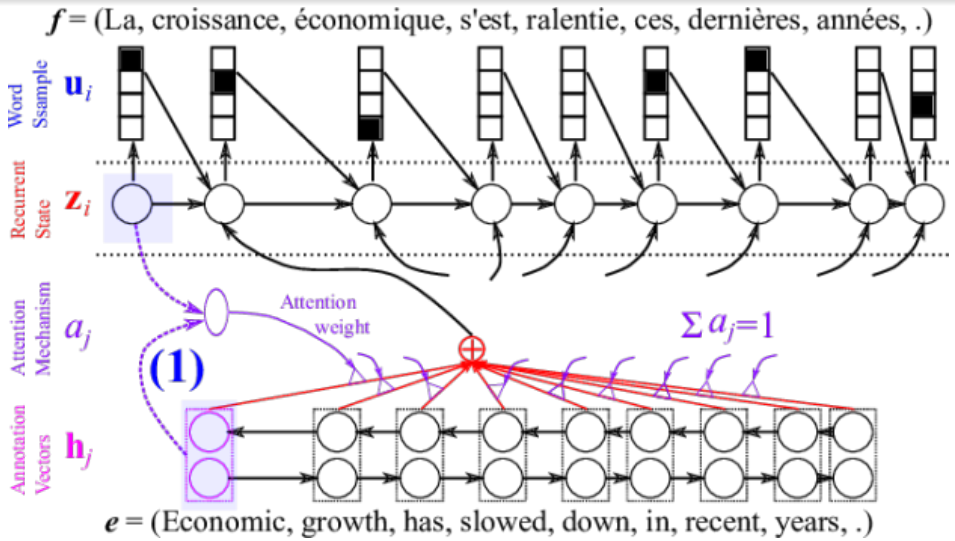


Kyunghyun Cho
http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

Attentional encoder-decoder

attention

- problem: how to incorporate variable-length context into hidden state?
- *attention model* computes *context vector* as weighted average of annotations
- weights are computed by feedforward neural network with softmax activation



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

Attentional encoder-decoder: math

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

- plain RNN instead of GRU
- simpler output layer
- we do not show bias terms

notation

- W, U, E, C, V are weight matrices (of different dimensionality)
  - E_x one-hot to embedding (e.g. $50000 \cdot 512$)
  - W_x embedding to hidden (e.g. $512 \cdot 1024$)
  - U_x hidden to hidden (e.g. $1024 \cdot 1024$)
  - C context (2x hidden) to hidden (e.g. $2048 \cdot 1024$)
  - ...
- 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: math

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: math

decoder

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

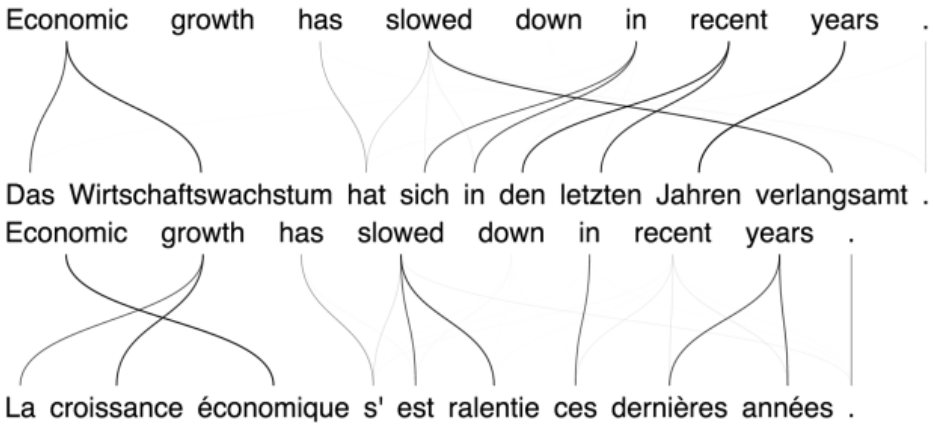
attention model

$$
\begin{aligned}
e_{ij} &= v_a^\top \tanh(W_a s_{i-1} + U_a h_j) \\
\alpha_{ij} &= \text{softmax}(e_{ij}) \\
c_i &= \sum_{j=1}^{T_x} \alpha_{ij} h_j
\end{aligned}
$$

Attention model

attention model

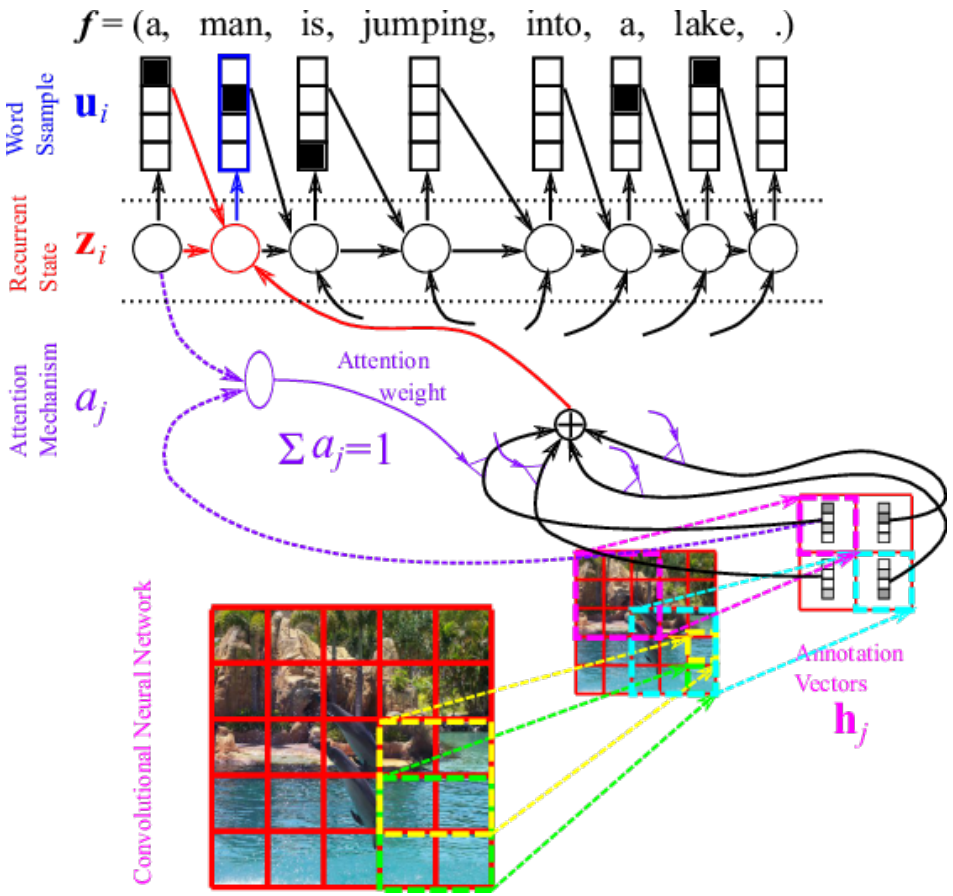
- side effect: we obtain alignment between source and target sentence
- applications:
  - visualisation
  - replace unknown words with back-off dictionary [Jean et al., 2015]
  - ...?



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

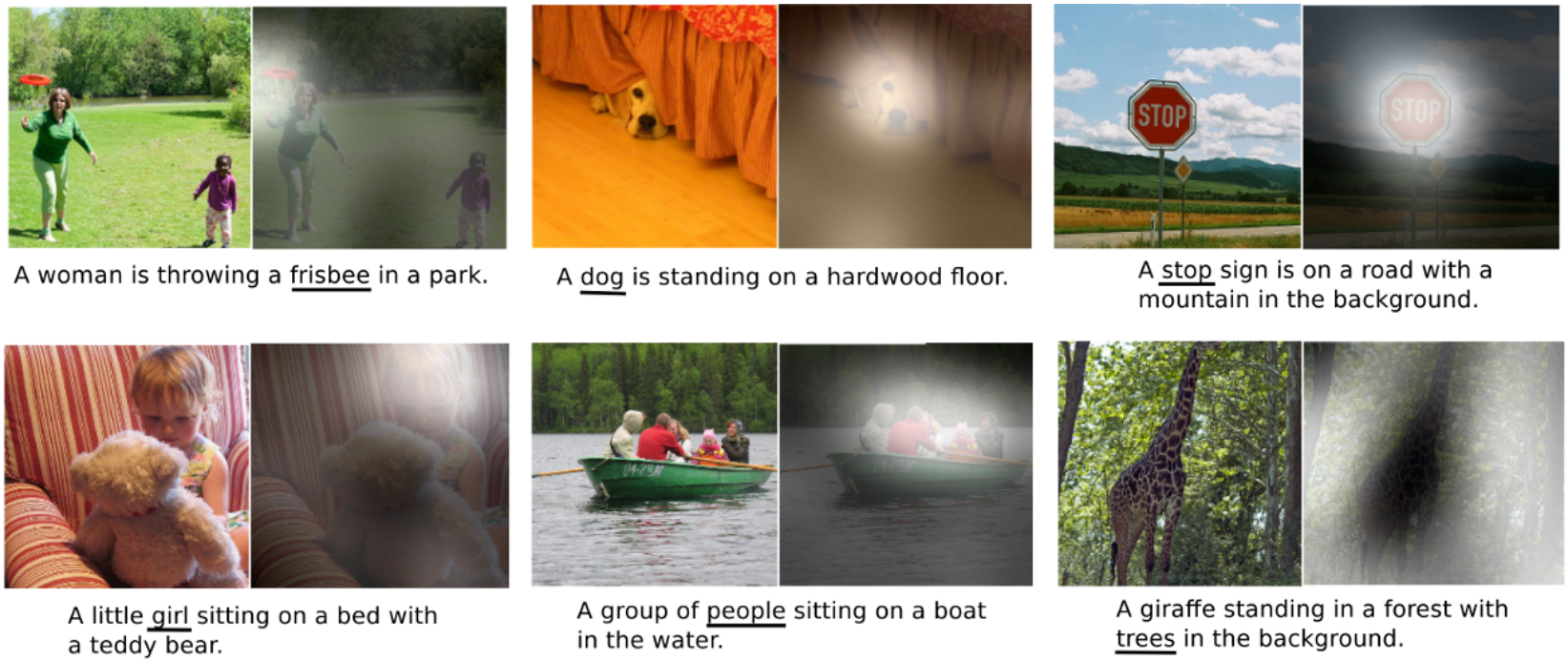


Fig. 5. Examples of the attention-based model attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word) [22]

[Cho et al., 2015]

Neural machine translation: decoding

decoding

- exact search intractable: $N^{|\text{vocab}|}$ for each possible output length N
  $\rightarrow$ approximative search for best translation
- given decoder state s_i, compute probability of each output word y_i
  - sampling: pick a random word (considering probability)
  - greedy search: pick the most probable word
  - beam search: pick the k most probable words, and compute y_{i+1} for each hypothesis in beam
- beam search with small beam ($k \approx 10$) seems sufficient for neural machine translation
- **ensemble**: compute probability distribution for next word with multiple models, and use (geometric) average

Further Reading

secondary literature

- lecture notes by Kyunghyun Cho: [Cho, 2015]
- introduction to LSTM (and GRU):

  `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`
- "Statistical Machine Translation" by Philipp Koehn (unpublished 2nd edition)

(A small selection of) Resources

feedforward neural LM toolkits

- CSLM `http://www-lium.univ-lemans.fr/cslm/`
- NPLM `https://github.com/moses-smt/nplm`
- OXLM `https://github.com/pauldb89/0xLM`

NMT tools

- dl4mt-tutorial (theano) `https://github.com/nyu-dl/dl4mt-tutorial`
  (our branch: nematus `https://github.com/rsennrich/nematus`)
- nmt.matlab `https://github.com/lmthang/nmt.matlab`
- seq2seq (tensorflow) `https://www.tensorflow.org/versions/r0.8/tutorials/seq2seq/index.html`

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/`
- please let me know about gaps in documentation
- NMT tools (previous slide) may also contain instructions/samples

Neural Machine Translation

1. neural network crash course

2. introduction to neural machine translation
   - neural language models
   - attentional encoder-decoder

3. recent research, opportunities and challenges in neural machine translation

State of Neural MT

- Attentional encoder-decoder networks are state of the art in MT
- similar models used for other NLP tasks

Attentional encoder-decoders (NMT) are SOTA

system	BLEU
uedin-nmt	34.2
metamind	32.3
NYU-UMontreal	30.8
cambridge	30.6
uedin-syntax	30.6
KIT/LIMSI	29.1
KIT	29.0
uedin-pbmt	28.4
jhu-syntax	26.6

Table: WMT16 results for EN→DE

system	BLEU
uedin-nmt	38.6
uedin-pbmt	35.1
jhu-pbmt	34.5
uedin-syntax	34.4
KIT	33.9
jhu-syntax	31.0

Table: WMT16 results for DE→EN

Attentional encoder-decoders (NMT) are SOTA

system	BLEU
uedin-nmt	34.2
metamind	32.3
NYU-UMontreal	30.8
cambridge	30.6
uedin-syntax	30.6
KIT/LIMSI	29.1
KIT	29.0
uedin-pbmt	28.4
jhu-syntax	26.6

Table: WMT16 results for EN→DE

system	BLEU
uedin-nmt	38.6
uedin-pbmt	35.1
jhu-pbmt	34.5
uedin-syntax	34.4
KIT	33.9
jhu-syntax	31.0

Table: WMT16 results for DE→EN

- pure NMT

Attentional encoder-decoders (NMT) are SOTA

system	BLEU
uedin-nmt	34.2
metamind	32.3
NYU-UMontreal	30.8
cambridge	30.6
uedin-syntax	30.6
KIT/LIMSI	29.1
KIT	29.0
uedin-pbmt	28.4
jhu-syntax	26.6

Table: WMT16 results for EN→DE

system	BLEU
uedin-nmt	38.6
uedin-pbmt	35.1
jhu-pbmt	34.5
uedin-syntax	34.4
KIT	33.9
jhu-syntax	31.0

Table: WMT16 results for DE→EN

- pure NMT
- NMT component

Attentional encoder-decoders (NMT) are SOTA

system	BLEU
uedin-nmt	34.2
metamind	32.3
NYU-UMontreal	30.8
cambridge	30.6
uedin-syntax	30.6
KIT/LIMSI	29.1
KIT	29.0
uedin-pbmt	28.4
jhu-syntax	26.6

Table: WMT16 results for EN→DE

system	BLEU
uedin-nmt	38.6
uedin-pbmt	35.1
jhu-pbmt	34.5
uedin-syntax	34.4
KIT	33.9
jhu-syntax	31.0

Table: WMT16 results for DE→EN

- pure NMT
- NMT component
- other neural components

Attentional encoder-decoders (NMT) are SOTA

uedin-nmt	25.8
NYU-UMontreal	23.6
jhu-pbmt	23.6
cu-chimera	21.0
uedin-cu-syntax	20.9
cu-tamchyna	20.8
cu-TectoMT	14.7
cu-mergedtrees	8.2

Table: WMT16 results for EN→CS

uedin-nmt	31.4
jhu-pbmt	30.4
PJATK	28.3
cu-mergedtrees	13.3

Table: WMT16 results for CS→EN

uedin-pbmt	35.2
uedin-nmt	33.9
uedin-syntax	33.6
jhu-pbmt	32.2
LIMSI	31.0

Table: WMT16 results for RO→EN

QT21-HimL-SysComb	28.9
uedin-nmt	28.1
RWTH-SYSCOMB	27.1
uedin-pbmt	26.8
uedin-lmu-hiero	25.9
KIT	25.8
lmu-cuni	24.3
LIMSI	23.9
jhu-pbmt	23.5
usfd-rescoring	23.1

Table: WMT16 results for EN→RO

Attentional encoder-decoders (NMT) are SOTA

uedin-nmt	26.0
amu-uedin	25.3
jhu-pbmt	24.0
LIMSI	23.6
AFRL-MITLL	23.5
NYU-UMontreal	23.1
AFRL-MITLL-verb-annot	20.9

Table: WMT16 results for EN→RU

amu-uedin	29.1
NRC	29.1
uedin-nmt	28.0
AFRL-MITLL	27.6
AFRL-MITLL-contrast	27.0

Table: WMT16 results for RU→EN

uedin-pbmt	23.4
uedin-syntax	20.4
PROMT-SMT	20.3
UH-factored	19.3
jhu-pbmt	19.1

Table: WMT16 results for FI→EN

abumatran-combo	17.4
abumatra-nmt	17.2
NYU-UMontreal	15.1
abumatran-pbsmt	14.6
jhu-pbmt	13.8
UH-factored	12.8
jhu-hltcoe	11.9
UUT	11.6
aalto	11.6

Table: WMT16 results for EN→FI

Selected examples from WMT16

system	sentence
source	Unsere digitalen Leben haben die Notwendigkeit, stark, lebenslustig und erfolgreich zu erscheinen, verdoppelt [...]
reference	Our digital lives have doubled the need to appear strong, fun-loving and successful [...]
uedin-nmt	Our digital lives have doubled the need to appear strong, lifelike and successful [...]
uedin-pbsmt	Our digital lives are lively, strong, and to be successful, doubled [...]

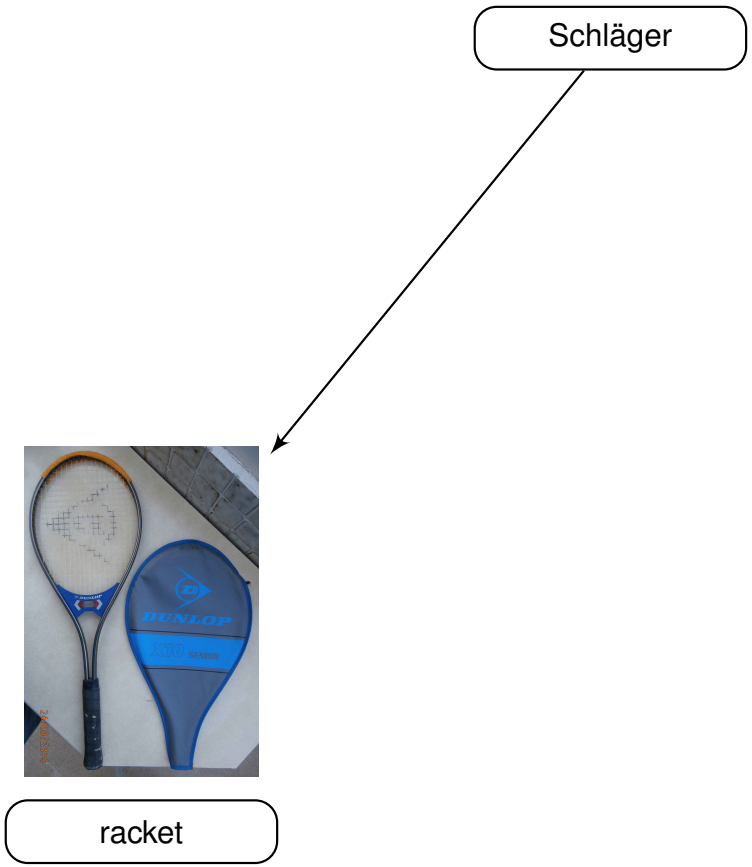
Selected examples from WMT16

system	sentence
source	Dort wurde er von dem Schläger und einer weiteren männlichen Person erneut angegriffen.
reference	There he was attacked again by his original attacker and another male.
uedin-nmt	There he was attacked again by the racket and another male person.
uedin-pbsmt	There, he was at the club and another male person attacked again.

Schläger

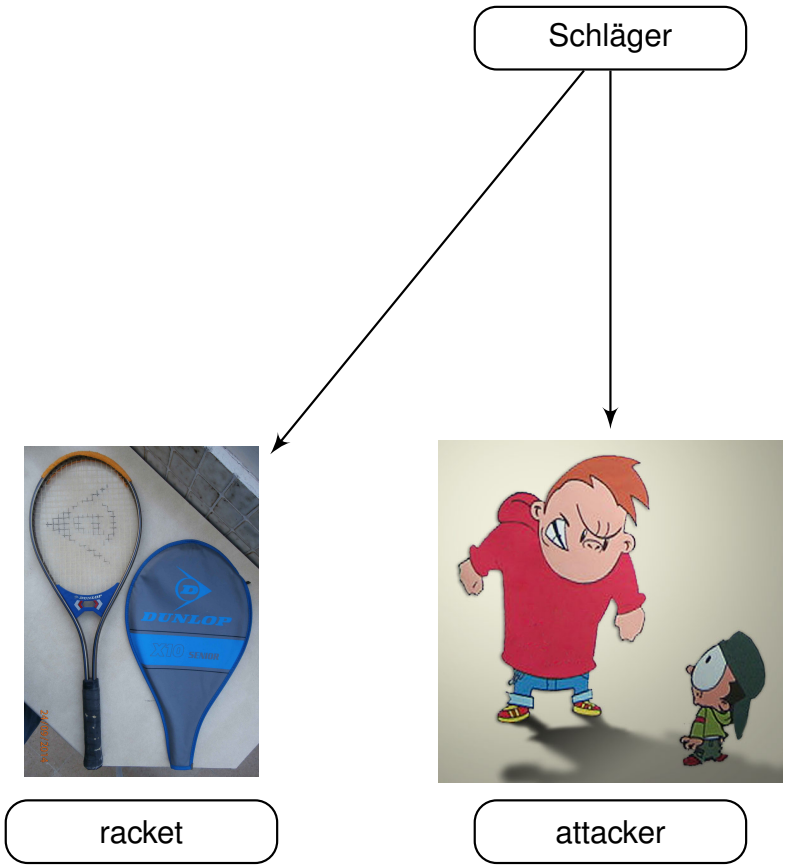
Selected examples from WMT16

system	sentence
source	Dort wurde er von dem Schläger und einer weiteren männlichen Person erneut angegriffen.
reference	There he was attacked again by his original attacker and another male.
uedin-nmt	There he was attacked again by the racket and another male person.
uedin-pbsmt	There, he was at the club and another male person attacked again.



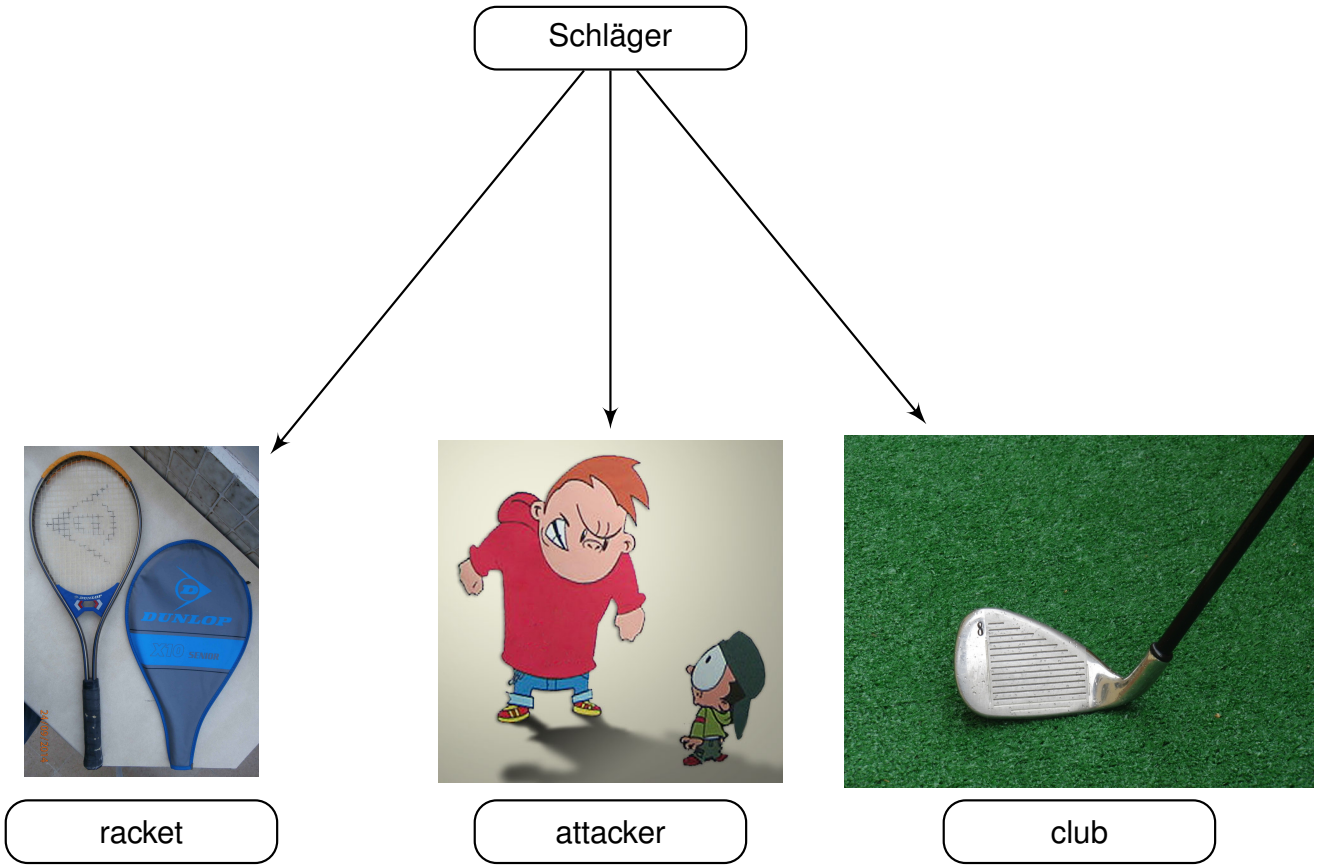
Selected examples from WMT16

system	sentence
source	Dort wurde er von dem Schläger und einer weiteren männlichen Person erneut angegriffen.
reference	There he was attacked again by his original attacker and another male.
uedin-nmt	There he was attacked again by the racket and another male person.
uedin-pbsmt	There, he was at the club and another male person attacked again.



Selected examples from WMT16

system	sentence
source	Dort wurde er von dem Schläger und einer weiteren männlichen Person erneut angegriffen.
reference	There he was attacked again by his original attacker and another male.
uedin-nmt	There he was attacked again by the racket and another male person.
uedin-pbsmt	There, he was at the club and another male person attacked again.



Selected examples from WMT16

system	sentence
source	Ein Jahr später machten die Fed-Repräsentanten diese Kürzungen rückgängig.
reference	A year later, Fed officials reversed those cuts.
uedin-nmt	A year later, FedEx officials reversed those cuts.
uedin-pbsmt	A year later, the Fed representatives made these cuts.

Selected examples from WMT16

system	sentence
source	Ein Jahr später machten die Fed-Repräsentanten diese Kürzungen rückgängig.
reference	A year later, Fed officials reversed those cuts.
uedin-nmt	A year later, FedEx officials reversed those cuts.
uedin-pbsmt	A year later, the Fed representatives made these cuts.

Selected examples from WMT16

system	sentence
source	Titelverteidiger ist Drittligaabsteiger SpVgg Unterhaching.
reference	The defending champions are SpVgg Unterhaching, who have been relegated to the third league.
uedin-nmt	Defending champion is third-round pick SpVgg Underhaching.
uedin-pbsmt	Title defender Drittligaabsteiger Week 2.

Selected examples from WMT16

system	sentence
source	Titelverteidiger ist Drittligaabsteiger SpVgg Unterhaching.
reference	The defending champions are SpVgg Unterhaching, who have been relegated to the third league.
uedin-nmt	Defending champion is third-round pick SpVgg Underhaching.
uedin-pbsmt	Title defender Drittligaabsteiger Week 2.

Comparison between phrase-based and neural MT

pro neural MT

- improved grammaticality [Neubig et al., 2015]
  - word order
  - insertion/deletion of function words
  - morphological agreement

pro phrase-based/syntax-based SMT

- minor degradation in lexical choice? [Neubig et al., 2015]

others

- rare/unseen words are problematic for both:
  - PBSMT suffers from data sparseness and noisy phrase alignment
  - (our) NMT system attempts subword-level translation

Why is neural MT output more grammatical?

neural MT

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

phrase-based SMT

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

Efficiency

speed bottlenecks

- matrix multiplication
  $\rightarrow$ use of highly parallel hardware (GPUs)
- softmax (scales with vocabulary size). Solutions:
  - LMs: hierarchical softmax; noise-contrastive estimation; self-normalization
  - NMT: approximate softmax through subset of vocabulary [Jean et al., 2015]

NMT training vs. decoding (on fast GPU)

- training: slow (1-3 weeks)
- decoding: fast (100 000–500 000 sentences / day)[a]

[a] with NVIDIA Titan X and amuNN (`https://github.com/emjotde/amunn`)

Open-vocabulary translation

Why is vocabulary size a problem?

- size of one-hot input/output vector is linear to vocabulary size
- large vocabularies are space inefficient
- large output vocabularies are time inefficient
- typical network vocabulary size: 30 000–100 000

What about out-of-vocabulary words?

- training set vocabulary typically larger than network vocabulary (1 million words or more)
- at translation time, we regularly encounter novel words:
  - names: *Barack Obama*
  - morph. complex words: *Hand|gepäck|gebühr* ('carry-on bag fee')
  - numbers, URLs etc.

Open-vocabulary translation

Solutions

- copy unknown words, or translate with back-off dictionary
  [Jean et al., 2015, Luong et al., 2015b, Gülçehre et al., 2016]
  $\rightarrow$ works for names (if alphabet is shared), and 1-to-1 aligned words
- use subword units (characters or others) for input/output vocabulary
  $\rightarrow$ model can learn translation of seen words on subword level
  $\rightarrow$ model can translate unseen words if translation is *transparent*

Transparent translations

Transparent translations

- some translations are semantically/phonologically transparent
  $\rightarrow$ no memorization needed; can be translated via subword units
- morphologically complex words (e.g. compounds):
  - solar system (English)
  - Sonnen|system (German)
  - Nap|rendszer (Hungarian)
- named entities:
  - Barack Obama (English; German)
  - Барак Обама (Russian)
  - バラク・オバマ (ba-ra-ku o-ba-ma) (Japanese)
- cognates and loanwords:
  - claustrophobia (English)
  - Klaustrophobie (German)
  - Клаустрофобия (Russian)
- many rare/unseen words belong to one of these categories

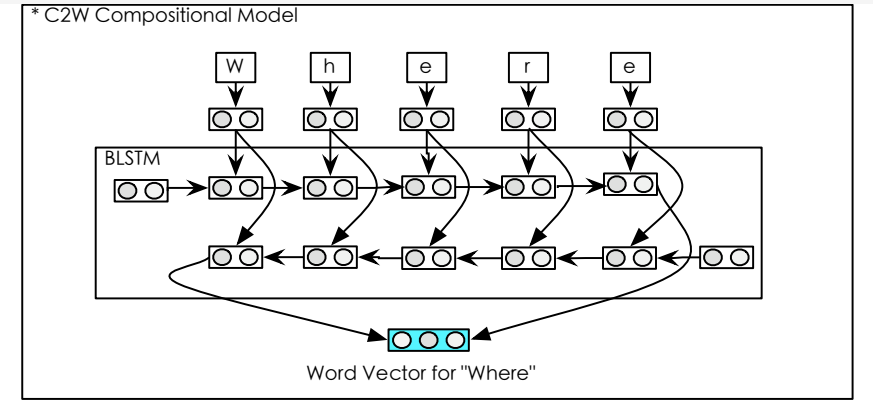
Subword neural machine translation

Flat representation [Sennrich et al., 2015b, Chung et al., 2016]

- sentence is a sequence of subword units

Hierarchical representation [Ling et al., 2015, Luong and Manning, 2016]

- sentence is a sequence of words
- words are a sequence of subword units



open question: should attention be on level of words or subwords?

Subword neural machine translation

Choice of subword unit

- character-level: small vocabulary, long sequences
- morphemes (?): hard to control vocabulary size
- hybrid choice: shortlist of words, subwords for rare words
- variable-length character n-grams: byte-pair encoding (BPE)

open research question which subword segmentation is best choice in terms of *efficiency* and *effectiveness*.

Byte pair encoding for word segmentation

word segmentation with byte-pair encoding [Sennrich et al., 2015b]

- actually a merge algorithm, starting from characters
- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: original vocabulary + one symbol per merge

'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

Byte pair encoding for word segmentation

word segmentation with byte-pair encoding [Sennrich et al., 2015b]

- actually a merge algorithm, starting from characters
- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: original vocabulary + one symbol per merge

'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

('e', 's') : 9

Byte pair encoding for word segmentation

word segmentation with byte-pair encoding [Sennrich et al., 2015b]

- actually a merge algorithm, starting from characters
- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: original vocabulary + one symbol per merge

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

('e', 's') : 9
('es', 't') : 9

Byte pair encoding for word segmentation

word segmentation with byte-pair encoding [Sennrich et al., 2015b]

- actually a merge algorithm, starting from characters
- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: original vocabulary + one symbol per merge

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

('e', 's') : 9
('es', 't') : 9
('est', '</w>') : 9

Byte pair encoding for word segmentation

word segmentation with byte-pair encoding [Sennrich et al., 2015b]

- actually a merge algorithm, starting from characters
- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: original vocabulary + one symbol per merge

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

('e', 's') : 9
('es', 't') : 9
('est', '</w>') : 9
('l', 'o') : 7

Byte pair encoding for word segmentation

word segmentation with byte-pair encoding [Sennrich et al., 2015b]

- actually a merge algorithm, starting from characters
- iteratively replace most frequent pair of symbols ('A','B') with 'AB'
- apply on dictionary, not on full text (for efficiency)
- output vocabulary: original vocabulary + one symbol per merge

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

('e', 's') : 9
('es', 't') : 9
('est', '</w>') : 9
('l', 'o') : 7
('lo', 'w') : 7
...

Byte pair encoding for word segmentation

why BPE?

- good trade-off between vocabulary size and text length

segmentation	# tokens	# types	# UNK
none	100 m	1 750 000	1079
characters	550 m	3000	0
BPE (60k operations)	112 m	63 000	0

- learned operations can be applied to unknown words
  $\rightarrow$ open-vocabulary

’l o w e s t </w>’

(’e’, ’s’) : 9
(’es’, ’t’) : 9
(’est’, ’</w>’) : 9
(’l’, ’o’) : 7
(’lo’, ’w’) : 7

Byte pair encoding for word segmentation

why BPE?

- good trade-off between vocabulary size and text length

segmentation	# tokens	# types	# UNK
none	100 m	1 750 000	1079
characters	550 m	3000	0
BPE (60k operations)	112 m	63 000	0

- learned operations can be applied to unknown words
  → open-vocabulary

’l o w es t </w>’

(’e’, ’s’) : 9
(’es’, ’t’) : 9
(’est’, ’</w>’) : 9
(’l’, ’o’) : 7
(’lo’, ’w’) : 7

Byte pair encoding for word segmentation

why BPE?

- good trade-off between vocabulary size and text length

segmentation	# tokens	# types	# UNK
none	100 m	1 750 000	1079
characters	550 m	3000	0
BPE (60k operations)	112 m	63 000	0

- learned operations can be applied to unknown words
  $\rightarrow$ open-vocabulary

’l o w est </w>’

(’e’, ’s’) : 9
(’es’, ’t’) : 9
(’est’, ’</w>’) : 9
(’l’, ’o’) : 7
(’lo’, ’w’) : 7

Byte pair encoding for word segmentation

why BPE?

- good trade-off between vocabulary size and text length

segmentation	# tokens	# types	# UNK
none	100 m	1 750 000	1079
characters	550 m	3000	0
BPE (60k operations)	112 m	63 000	0

- learned operations can be applied to unknown words
  → open-vocabulary

’l o w est</w>’

(’e’, ’s’) : 9
(’es’, ’t’) : 9
(’est’, ’</w>’) : 9
(’l’, ’o’) : 7
(’lo’, ’w’) : 7

Byte pair encoding for word segmentation

why BPE?

- good trade-off between vocabulary size and text length

segmentation	# tokens	# types	# UNK
none	100 m	1 750 000	1079
characters	550 m	3000	0
BPE (60k operations)	112 m	63 000	0

- learned operations can be applied to unknown words
  $\rightarrow$ open-vocabulary

	('e', 's') : 9
	('es', 't') : 9
'lo w est</w>'	('est', '</w>') : 9
	('l', 'o') : 7
	('lo', 'w') : 7

Byte pair encoding for word segmentation

why BPE?

- good trade-off between vocabulary size and text length

segmentation	# tokens	# types	# UNK
none	100 m	1 750 000	1079
characters	550 m	3000	0
BPE (60k operations)	112 m	63 000	0

- learned operations can be applied to unknown words
  $\rightarrow$ open-vocabulary

'low est</w>'

('e', 's') : 9
('es', 't') : 9
('est', '</w>') : 9
('l', 'o') : 7
('lo', 'w') : 7

Byte pair encoding for word segmentation

system	sentence
source	health research institutes
reference	Gesundheitsforschungsinstitute
copy unknown words	Forschungsinstitute
character bigrams	Fo|rs|ch|un|gs|in|st|it|ut|io|ne|n
BPE (joint vocabulary)	Gesundheits|forsch|ungsin|stitute
source	asinine situation
reference	dumme Situation
copy unknown words	asinine situation → UNK → asinine
character bigrams	as|in|in|e situation → As|in|en|si|tu|at|io|n
BPE (joint vocabulary)	as|in|ine situation → As|in|in-|Situation

Table: English→German translation example. "|" marks subword boundaries.

system	sentence
source	Mirzayeva
reference	Мирзаева (Mirzaeva)
copy unknown words	Mirzayeva → UNK → Mirzayeva
character bigrams	Mi|rz|ay|ev|a → Ми|рз|ае|ва (Mi|rz|ae|va)
BPE (joint vocabulary)	Mir|za|yeva → Мир|за|ева (Mir|za|eva)

Table: English→Russian translation example. "|" marks subword boundaries.

Architecture variants

- convolution network as encoder [Kalchbrenner and Blunsom, 2013]
- TreeLSTM as encoder [Eriguchi et al., 2016]
- modifications to attention mechanism
  [Luong et al., 2015a, Feng et al., 2016, Tu et al., 2016, Mi et al., 2016]
  $\rightarrow$ goal of better modelling distortion, coverage, etc.
- reward symmetry between source-to-target and target-to-source attention [Cohn et al., 2016, Cheng et al., 2015]

Sequence-level training

- problem: at training time, target-side history is reliable; at test time, it is not.
- solution: instead of using gold context, sample from the model to obtain target context
  [Shen et al., 2015, Ranzato et al., 2015, Bengio et al., 2015]
- more efficient cross entropy training remains in use to initialize weights

Training data: monolingual

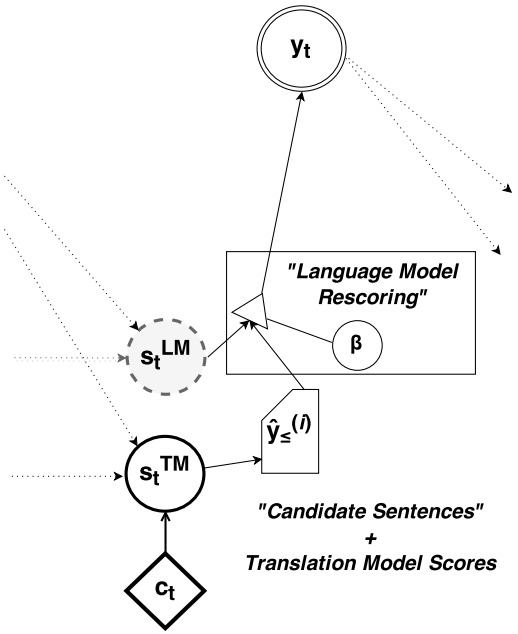
Why train on monolingual data?

- cheaper to create/collect
- parallel data is scarce for many language pairs
- domain adaptation with *in-domain* monolingual data

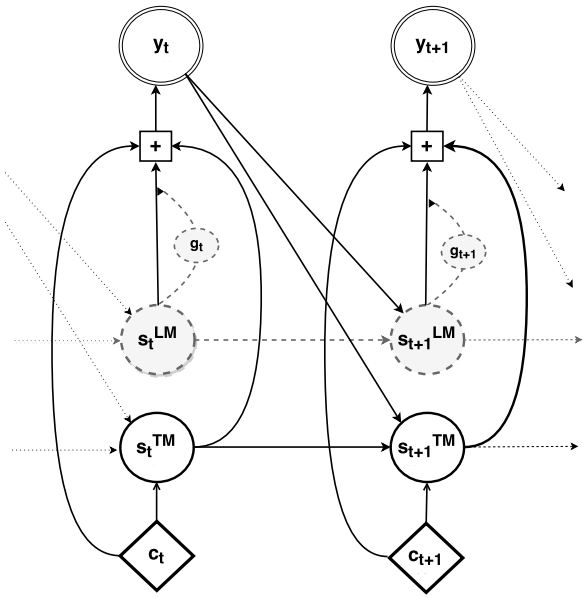
Training data: monolingual

Solutions/1

- shallow fusion: rerank with language model [Gülçehre et al., 2015]
- deep fusion: extra, LM-specific hidden layer [Gülçehre et al., 2015]



(a) Shallow Fusion (Sec. 4.1)



(b) Deep Fusion (Sec. 4.2)

[Gülçehre et al., 2015]

Training data: monolingual

Solutions/2

- decoder is already a language model. Train encoder-decoder with added monolingual data [Sennrich et al., 2015a]

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

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

- how do we get approximation of context vector c_i?
  - dummy source context (moderately effective)
  - automatically back-translate monolingual data into source language

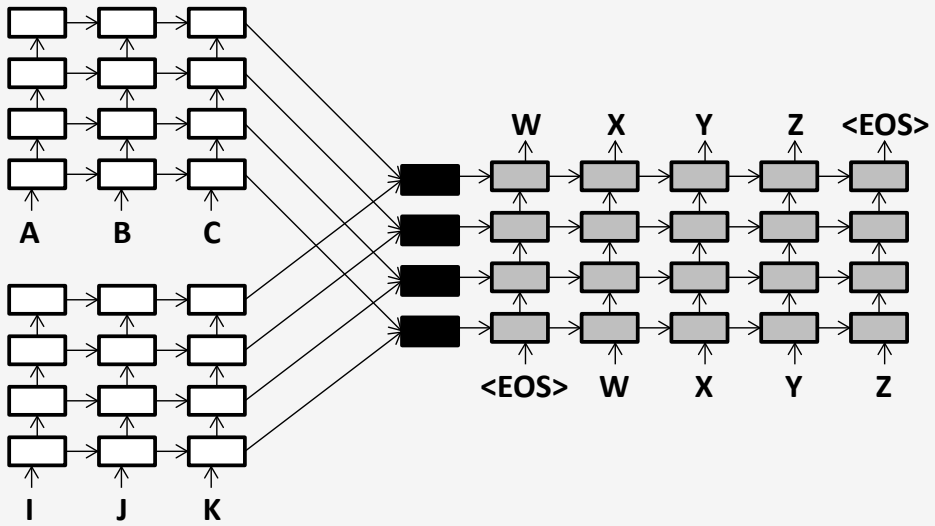
name	2014	2015
PBSMT [Haddow et al., 2015]	28.8	29.3
NMT [Gülçehre et al., 2015]	23.6	-
shallow fusion [Gülçehre et al., 2015]	23.7	-
deep fusion [Gülçehre et al., 2015]	24.0	-
NMT baseline	25.9	26.7
+back-translated monolingual data	**29.5**	**30.4**

Table: DE→EN translation performance (BLEU) on WMT training/test sets.

Training data: multilingual

Multi-source translation [Zoph and Knight, 2016]

we can condition on multiple input sentences

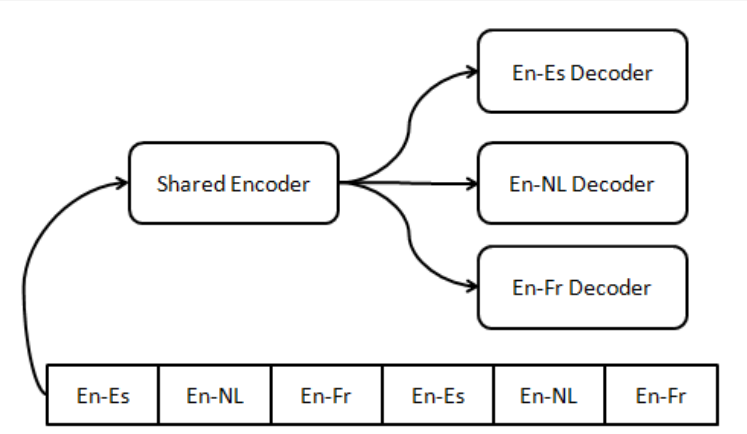


- benefits:
  - one source text may contain information that is undespecified in other
    $\rightarrow$ possible quality gains
- drawbacks:
  - we need multiple source sentences at training and decoding time

Training data: multilingual

Multilingual models [Dong et al., 2015, Firat et al., 2016]

we can share layers of the model across language pairs

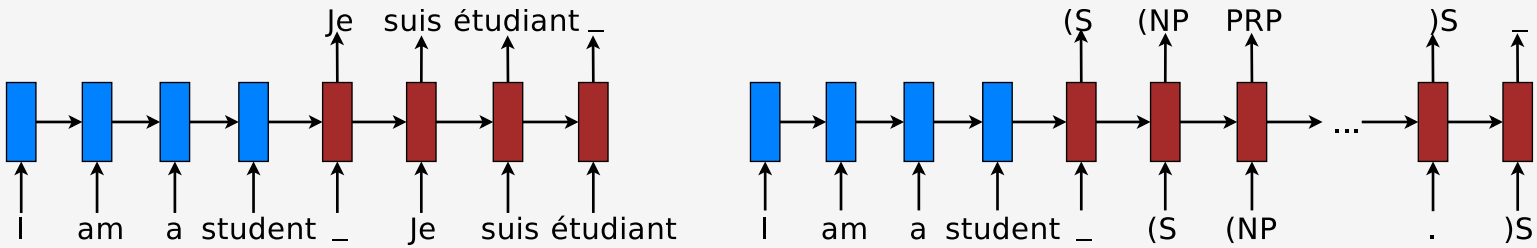


- benefits:
  - transfer learning from one language pair to the other
    $\rightarrow$ possible quality gains, especially for low-resourced language pairs
  - scalability: do we need $N^2 - N$ independent models for N languages?
    $\rightarrow$ sharing of parameters allows linear growth
- drawbacks:
  - generalization to language pairs with no training data untested

Training data: other tasks

Multi-task models [Luong et al., 2016]

- other tasks can be modelled with sequence-to-sequence models
- we can share layers between translation and other tasks



NMT as a component in log-linear models

Log-linear models

- model ensembling is well-established
- reranking output of phrase-based/syntax-based with NMT [Neubig et al., 2015]
- incorporating NMT as a feature function into PBSMT [Junczys-Dowmunt et al., 2016]
  $\rightarrow$ results depend on relative performance of PBSMT and NMT
- log-linear combination of different neural models
  - left-to-right and right-to-left [Liu et al., 2016]
  - source-to-target and target-to-source [Li and Jurafsky, 2016]

Some future directions for (neural) MT research

- (better) solutions to new(ish) problems
  - OOVs, coverage, efficiency...
- new solutions to old problems
  - consider context beyond sentence boundary
  - reward semantic adequacy of translation
  - deal with underspecified input
    - Chinese tense
    - Spanish zero pronouns
    - English politeness
- new opportunities
  - one model for many language pairs?
  - tight integration with other NLP tasks

Lab session this afternoon

- hands-on session with loose guidance
- theano (+CUDA) installation session
- train your own NMT model
  `https://github.com/rsennrich/wmt16-scripts`
- try decoding with existing model
  `http://statmt.org/rsennrich/wmt16_systems/`

Thank you!

Bibliography I

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, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015).

Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks.

CoRR, abs/1506.03099.

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

A Neural Probabilistic Language Model.

J. Mach. Learn. Res., 3:1137–1155.

Cheng, Y., Shen, S., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2015).

Agreement-based Joint Training for Bidirectional Attention-based Neural Machine Translation.

CoRR, abs/1512.04650.

Cho, K. (2015).

Natural Language Understanding with Distributed Representation.

CoRR, abs/1511.07916.

Cho, K., Courville, A., and Bengio, Y. (2015).

Describing Multimedia Content using Attention-based Encoder-Decoder Networks.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014).

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.

In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Bibliography II

Chung, J., Cho, K., and Bengio, Y. (2016).

A Character-level Decoder without Explicit Segmentation for Neural Machine Translation.
CoRR, abs/1603.06147.

Cohn, T., Hoang, C. D. V., Vymolova, E., Yao, K., Dyer, C., and Haffari, G. (2016).

Incorporating Structural Alignment Biases into an Attentional Neural Translation Model.
CoRR, abs/1601.01085.

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.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015).

Multi-Task Learning for Multiple Language Translation.
In
Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference o
pages 1723–1732, Beijing, China. Association for Computational Linguistics.

Eriguchi, A., Hashimoto, K., and Tsuruoka, Y. (2016).

Tree-to-Sequence Attentional Neural Machine Translation.
ArXiv e-prints.

Feng, S., Liu, S., Li, M., and Zhou, M. (2016).

Implicit Distortion and Fertility Models for Attention-based Encoder-Decoder NMT Model.
CoRR, abs/1601.03317.

Bibliography III


Firat, O., Cho, K., and Bengio, Y. (2016).
Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism.
ArXiv e-prints.


Gülçehre, c., Ahn, S., Nallapati, R., Zhou, B., and Bengio, Y. (2016).
Pointing the Unknown Words.
CoRR, abs/1603.08148.


Gülçehre, c., Firat, O., Xu, K., Cho, K., Barrault, L., Lin, H., Bougares, F., Schwenk, H., and Bengio, Y. (2015).
On Using Monolingual Corpora in Neural Machine Translation.
CoRR, abs/1503.03535.


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.


Junczys-Dowmunt, M., Dwojak, T., and Sennrich, R. (2016).
The AMU-UEDIN Submission to the WMT16 News Translation Task: Attention-based NMT Models as Feature Functions in Phrase-based SMT.
In First Conference on Machine Translation (WMT16), Berlin, Germany.

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.

Li, J. and Jurafsky, D. (2016).

Mutual Information and Diverse Decoding Improve Neural Machine Translation.
CoRR, abs/1601.00372.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015).

Character-based Neural Machine Translation.
ArXiv e-prints.

Liu, L., Utiyama, M., Finch, A., and Sumita, E. (2016).

Agreement on Target-bidirectional Neural Machine Translation .
In NAACL HLT 16, San Diego, CA.

Luong, M., Le, Q. V., Sutskever, I., Vinyals, O., and Kaiser, L. (2016).

Multi-task Sequence to Sequence Learning.
In ICLR 2016.

Luong, M.-T. and Manning, C. D. (2016).

Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models.
ArXiv e-prints.

Luong, T., Pham, H., and Manning, C. D. (2015a).

Effective Approaches to Attention-based Neural Machine Translation.
In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Bibliography V

Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015b).
Addressing the Rare Word Problem in Neural Machine Translation.
In
Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference o
pages 11–19, Beijing, China. Association for Computational Linguistics.

Mi, H., Sankaran, B., Wang, Z., and Ittycheriah, A. (2016).
A Coverage Embedding Model for Neural Machine Translation.
ArXiv e-prints.

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.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015).
Sequence Level Training with Recurrent Neural Networks.
CoRR, abs/1511.06732.

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., Haddow, B., and Birch, A. (2015a).
Improving Neural Machine Translation Models with Monolingual Data.
ArXiv e-prints.

Sennrich, R., Haddow, B., and Birch, A. (2015b).
Neural Machine Translation of Rare Words with Subword Units.
CoRR, abs/1508.07909.

Bibliography VI


Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2015).
Minimum Risk Training for Neural Machine Translation.
CoRR, abs/1512.02433.


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.


Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. (2016).
Coverage-based Neural Machine Translation.
CoRR, abs/1601.04811.


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.


Zoph, B. and Knight, K. (2016).
Multi-Source Neural Translation.
In NAACL HLT 2016.

Hands-on session: install theano

- theano depends on bleeding-edge numpy
- my suggestion: to avoid version conflicts, install in Python virtual environment
  - pip install –user virtualenv #install virtualenv
  - virtualenv virtual_environment #create environemnt
  - source virtual_environment/bin/activate #start environment
  - pip install numpy numexpr cython tables theano ipdb #install theano
- you may need to install BLAS library and other dependencies
  on Debian Linux:
  sudo apt-get install liblapack-dev libblas-dev gfortran libhdf5-serial-dev
- if you have NVIDIA GPU, you should install CUDA
  if you don't, training will be too slow

Hands-on session: train your own model

- sample scripts and config at
  `https://github.com/rsennrich/wmt16-scripts`
- requirements:
  - mosesdecoder (just for preprocessing, no installation required)
    git clone https://github.com/moses-smt/mosesdecoder
  - subword-nmt (for BPE segmentation)
    git clone https://github.com/rsennrich/subword-nmt
  - Nematus (DL4MT fork; for training NMT)
    git clone https://www.github.com/rsennrich/nematus
- set paths in shell scripts, then execute preprocess.sh, then train.sh
- to train actual model, use more data, and be patient
  script prints status after every 1000 minibatches
  $\rightarrow \approx$ 30 min if CUDA is set up properly
  (we train for 300000–600000 minibatches)

Hands-on session: use pre-trained model

- download model(s) from
  `http://statmt.org/rsennrich/wmt16_systems/`
- requirements:
  - mosesdecoder (just for preprocessing, no installation required)
    git clone https://github.com/moses-smt/mosesdecoder
  - subword-nmt (for BPE segmentation)
    git clone https://github.com/rsennrich/subword-nmt
  - Nematus (DL4MT fork; for decoding)
    git clone https://www.github.com/rsennrich/nematus
- set paths in translate.sh, then execute:
  echo "This is a test." | ./translate.sh