Neural Machine Translation

Antonio Valerio Miceli Barone

The University of Edinburgh

September 1, 2019

Language modeling

autoregressive language modeling

- language modeling: estimate the probability distribution of sentences
- a sentence T of length n is a sequence of tokens $w_1, \ldots, w_n \in [1, \ldots, K]$
- autoregressive decomposition: estimate the probability of each token given its prefix

$$
\begin{aligned}
p(T) &= p(w_1, \ldots, w_n) \\
&= \prod_{i=1}^{n} p(w_i | w_1, \ldots, w_{i-1}) \qquad \text{(chain rule)}
\end{aligned}
$$

Language modeling with neural networks

neural network probability estimator

$$p(w_i = k | w_1, \ldots, w_{i-1}) = f_k(w_1, \ldots, w_{i-1}; \theta)$$

- f is a neural network with parameters θ that computes a vector of K probabilities

Language modeling with neural networks

neural network probability estimator

$$p(w_i = k | w_1, \ldots, w_{i-1}) = f_k(w_1, \ldots, w_{i-1}; \theta)$$

- f is a neural network with parameters θ that computes a vector of K probabilities
- general structure:

$$f(w_1, \ldots, w_{i-1}) = \text{softmax}(Proj(Seq(Emb(w_1), \ldots, Emb(w_{i-1}))))$$

  - $Emb(k) = W^{Emb}_{:,k}$: word embeddings $W^{Emb} \in \mathcal{R}^{d \times K}$
  - $Seq(x_1, \ldots, x_{i-1})$: sequence combinator
  - $Proj(s) = W^{Out} \cdot s$: output projection $W^{Out} \in \mathcal{R}^{K \times d}$.

Language modeling with neural networks

neural network probability estimator

$$p(w_i = k | w_1, \ldots, w_{i-1}) = f_k(w_1, \ldots, w_{i-1}; \theta)$$

- f is a neural network with parameters θ that computes a vector of K probabilities
- general structure:

$$f(w_1, \ldots, w_{i-1}) = \text{softmax}(Proj(Seq(Emb(w_1), \ldots, Emb(w_{i-1}))))$$

  - $Emb(k) = W^{Emb}_{:,k}$: word embeddings $W^{Emb} \in \mathcal{R}^{d \times K}$
  - $Seq(x_1, \ldots, x_{i-1})$: sequence combinator
  - $Proj(s) = W^{Out} \cdot s$: output projection $W^{Out} \in \mathcal{R}^{K \times d}$.
    - usually $W^{Out} = \text{transpose}(W^{Emb})$ [Press and Wolf, 2017]

Training

Maximum Likelihood Estimation

- maximize the log-likelihood of the training set $\{T^{(j)}\}$ under the model

$$\underset{\theta}{\operatorname{argmax}} \sum_{j} \sum_{i=1}^{n^{(j)}} \log p(w_i^{(j)} | w_1^{(j)}, \ldots, w_{i-1}^{(j)})$$

- mini-batch stochastic gradient descent
- adaptive learning rate and momentum (e.g. Adam optimizer)

Training

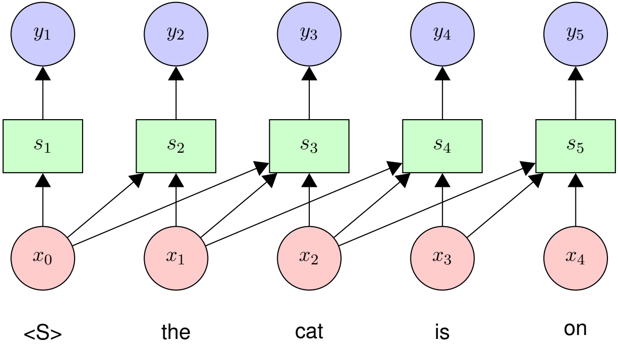
Maximum Likelihood Estimation

- maximize the log-likelihood of the training set $\{T^{(j)}\}$ under the model

$$\underset{\theta}{\operatorname{argmax}} \sum_{j} \sum_{i=1}^{n^{(j)}} \log p(w_i^{(j)} | w_1^{(j)}, \ldots, w_{i-1}^{(j)})$$

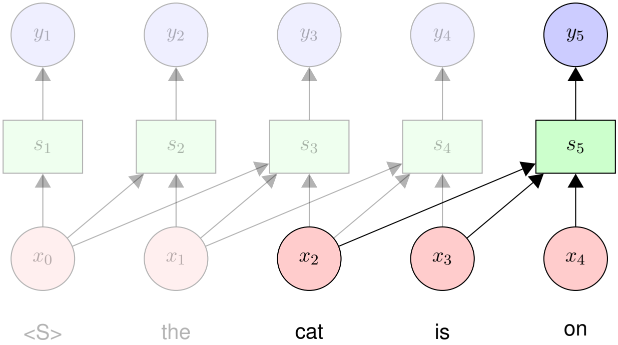
- mini-batch stochastic gradient descent
- adaptive learning rate and momentum (e.g. Adam optimizer)
- other training criteria can be used (e.g. reinforcement learning, GANs)
    - in practice it's hard to do better than MLE

Convolutional language model [Bengio et al., 2003]



Fixed width sliding window of lenth L

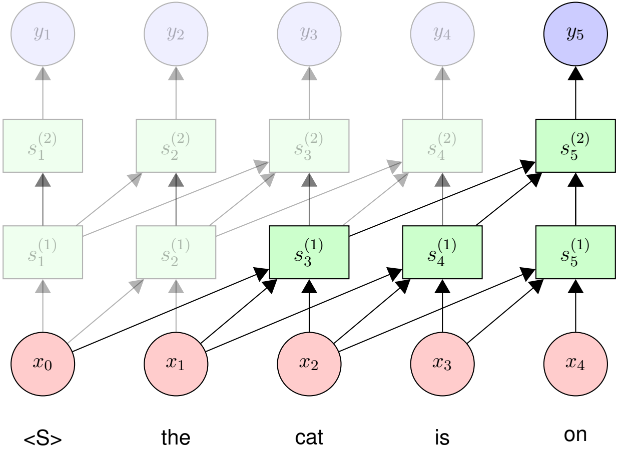
Convolutional language model



Fixed width sliding window of lenth L

$$Seq(x_1, \ldots, x_{i-1}) = Seq(x_{i-L}, \ldots, x_{i-1})$$
$$= \text{ReLU}(b^{conv} + \sum_{j=1}^{L} W^{conv}_{:,:,j} \cdot x_{i-j})$$

Convolutional language model



Multiple layers increase both depth and window size

Convolutional language model

pros & cons

- pro: training can be parallelized over words
- con: strong Markovian independence assumption

Convolutional language model

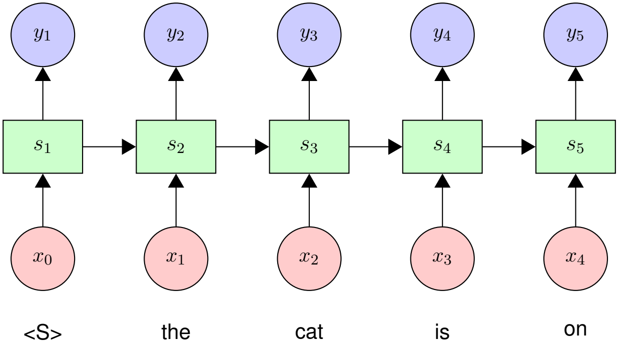
pros & cons

- pro: training can be parallelized over words
- con: strong Markovian independence assumption

extensions [Bai et al., 2018]

- residual connections
- normalization layers (e.g. batch norm, layer norm)
- dilated convolutions

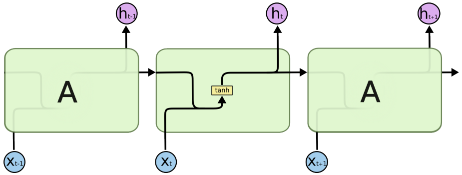
Recurrent language model [Mikolov et al., 2010]



Recurrent decomposition

$$\begin{aligned} Seq(x_1, \ldots, x_{i-1}) &= \text{RNN}(Seq(x_1, \ldots, x_{i-2}), x_{i-1}) \\ s_0 &= 0 \\ s_i &= \text{RNN}(s_{i-1}, x_{i-1}) \end{aligned}$$

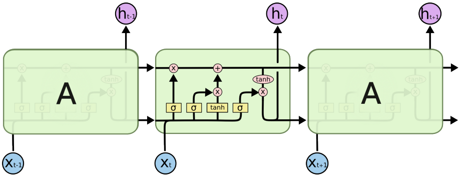
RNN variants



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)

Recurrent language model

pros & cons

- pro: can capture long distance dependencies
- pro: can represent arbitrary FSAs (+ counting)
- con: inherently sequential even during training
- con: hidden state can become a bottleneck

Recurrent language model

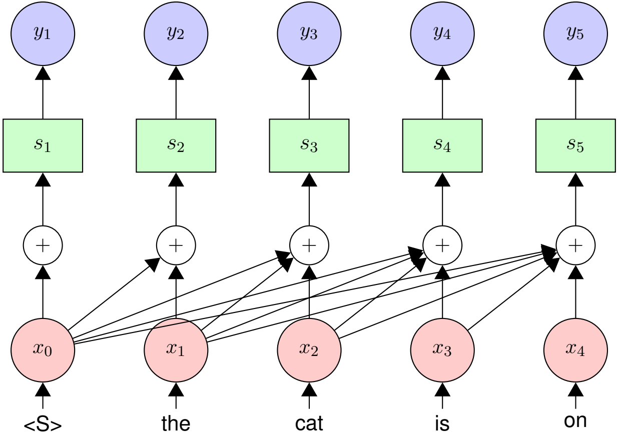
pros & cons

- pro: can capture long distance dependencies
- pro: can represent arbitrary FSAs (+ counting)
- con: inherently sequential even during training
- con: hidden state can become a bottleneck

extensions

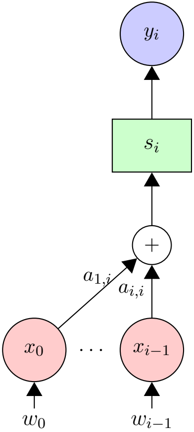
- stacked depth and transition depth [Miceli Barone et al., 2017]
- residual connections
- normalization layers (layer norm)
- etc.

Transformer language model [Vaswani et al., 2017]



Causal self-attention

Transformer language model



causal self-attention

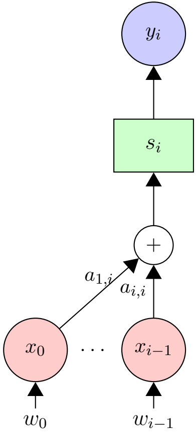
$$e_{j,i} = x_{j-1}^{\dagger} \cdot W^K \cdot W^Q \cdot x_{i-1} \text{ (dot product attention)}$$

$$a_{j,i} = \operatorname*{softmax}_{j \leq i}(e_{j,i})$$

$$c_i = \sum_{j=1}^{i} a_{j,i} W^V \cdot x_{j-1}$$

$$s_i = b^{(2)} + W^{(2)} \cdot \text{ReLU}(b^{(1)} + W^{(1)} \cdot c_i)$$

Transformer language model



causal self-attention

$$e_{j,i} = x_{j-1}^{\dagger} \cdot W^K \cdot W^Q \cdot x_{i-1} \text{ (dot product attention)}$$

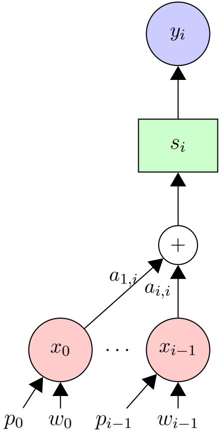
$$a_{j,i} = \underset{j \leq i}{\text{softmax}}(e_{j,i})$$

$$c_i = \sum_{j=1}^{i} a_{j,i} W^V \cdot x_{j-1}$$

$$s_i = b^{(2)} + W^{(2)} \cdot \text{ReLU}(b^{(1)} + W^{(1)} \cdot c_i)$$

what about word order?

Transformer language model



causal self-attention

$$e_{j,i} = x_{j-1}^{\dagger} \cdot W^K \cdot W^Q \cdot x_{i-1} \text{ (dot product attention)}$$

$$a_{j,i} = \underset{j \leq i}{\text{softmax}}(e_{j,i})$$

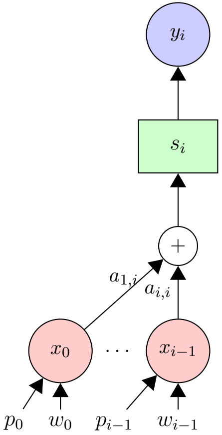
$$c_i = \sum_{j=1}^{i} a_{j,i} W^V \cdot x_{j-1}$$

$$s_i = b^{(2)} + W^{(2)} \cdot \text{ReLU}(b^{(1)} + W^{(1)} \cdot c_i)$$

what about word order?

$x_i = x_i^{word} + x_i^{pos}$

Transformer language model



transformer in practice

$$x_i = x_i^{word} + x_i^{pos} \text{ (position embedding)}$$

$$e_{j,i}^{(h)} = x_{j-1}^{\dagger} \cdot W^{K_h} \cdot W^{Q_h} \cdot x_{i-1}$$

$$a_{j,i}^{(h)} = \operatorname*{softmax}_{j \leq i}(e_{j,i}^{(h)})$$

$$c_i^{(h)} = \sum_{j=1}^{i} a_{j,i}^{(h)} W^{V_h} \cdot x_{j-1}$$

$$\widetilde{c}_i = \operatorname*{concat}_{h}(c_i^{(h)}) \text{ (multi-head attention)}$$

$$c_i = \text{layerNorm}(x_i + \widetilde{c}_i) \text{ (residual and layernorm)}$$

$$\widetilde{s}_i = b^{(2)} + W^{(2)} \cdot \text{ReLU}(b^{(1)} + W^{(1)} \cdot c_i)$$

$$s_i = \text{layerNorm}(c_i + \widetilde{s}_i) \text{ (residual and layernorm)}$$

Transformer language model

pros & cons

- pro: **SOTA on everything**
- pro: can capture long distance dependencies
- pro: training can be parallelized over words
- con: theoretical complexity increases at each step
  - not an issue for single sentences
- con: tricky to train
  - require dropout, learning rate warmup, label smoothing, etc.

Transformer language model

pros & cons

- pro: **SOTA on everything**
- pro: can capture long distance dependencies
- pro: training can be parallelized over words
- con: theoretical complexity increases at each step
  - not an issue for single sentences
- con: tricky to train
  - require dropout, learning rate warmup, label smoothing, etc.

extensions

- Transformer-XL [Dai et al., 2019]
  - recurrent over sentences
- dynamic convolutions [Wu et al., 2019]
- etc.

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:

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

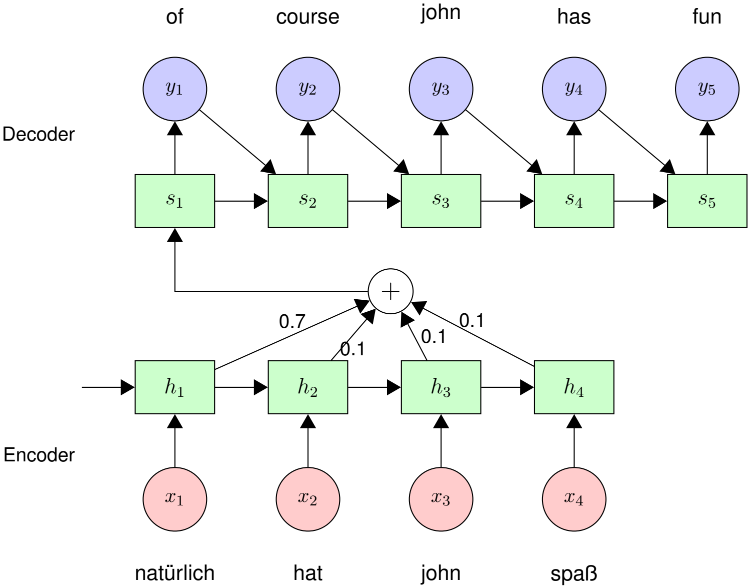
- Translation model:

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

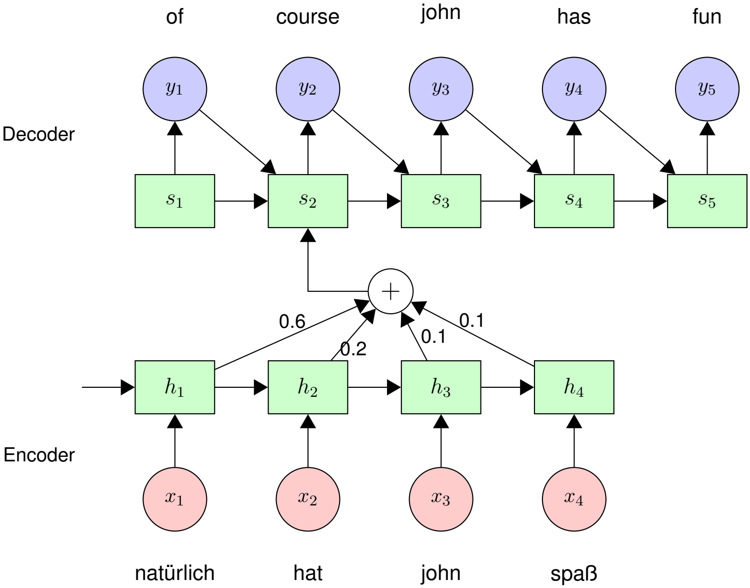
- We could just treat sentence pair as one long sequence, but:
  - We do not care about $p(S)$
  - We may want different vocabulary, network architecture for source text

$\rightarrow$ Use separate neural networks for source and target with an attention mechanism

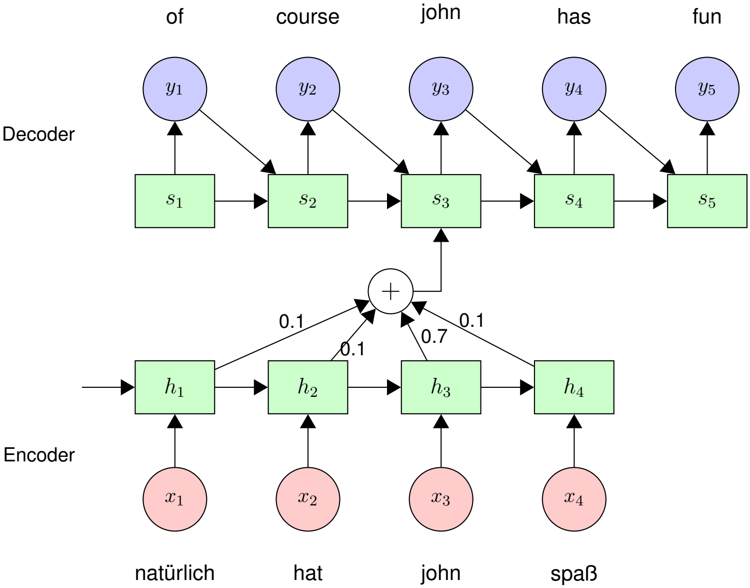
Recurrent Encoder-Decoder with Attention



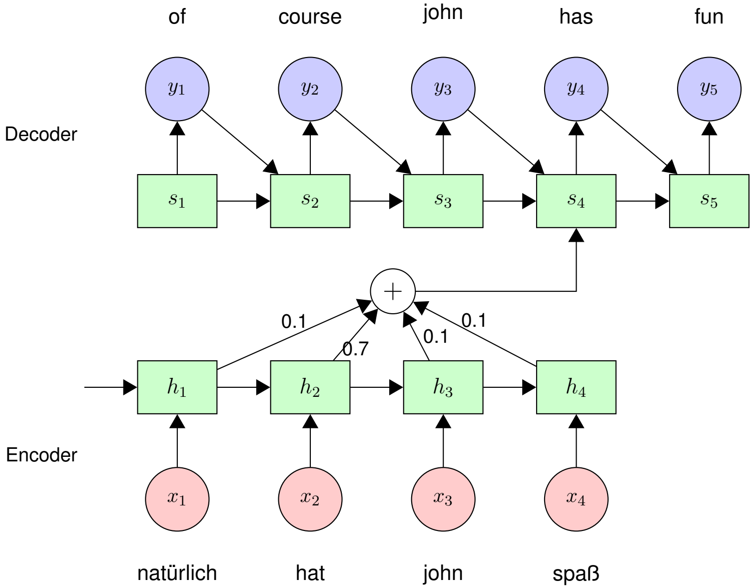
Recurrent Encoder-Decoder with Attention



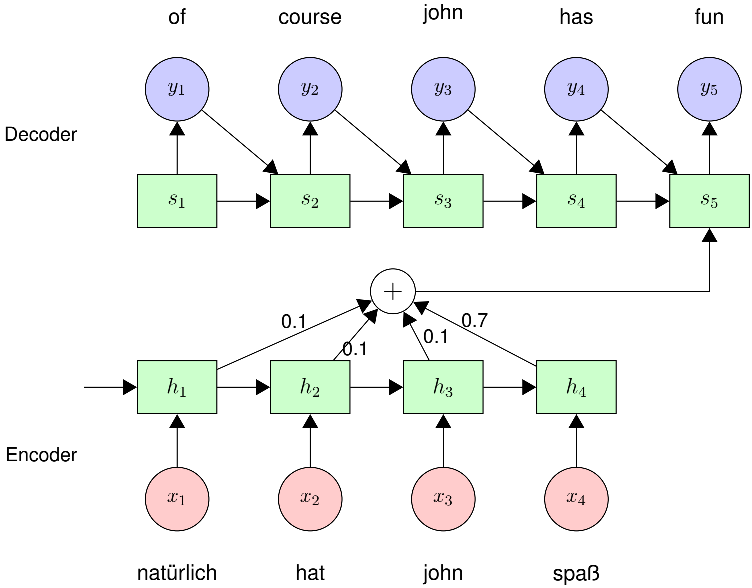
Recurrent Encoder-Decoder with Attention



Recurrent Encoder-Decoder with Attention



Recurrent Encoder-Decoder with Attention



Recurrent Attentional encoder-decoder: Maths

encoder

$$\overrightarrow{h}_j = \begin{cases} 0, & \text{, if } j = 0 \\ \text{RNN}(h_{j-1}, x_j) & \text{, if } j > 0 \end{cases}$$

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

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

Recurrent Attentional encoder-decoder: Maths

decoder

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

$$t_i = \mathsf{tanh}(U_o s_i + W^{out} E_y y_{i-1} + C_o c_i)$$

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

cross-attention

$$e_{i,j} = h_j^\dagger \cdot W^K \cdot W^Q \cdot s_{i-1}$$

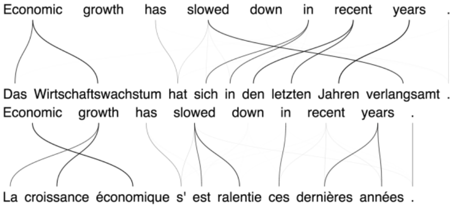
$$a_{i,j} = \mathsf{softmax}_j(e_{i,j})$$

$$c_i = \sum_{j=1}^{T_x} a_{i,j} W^V \cdot h_j$$

Attention model

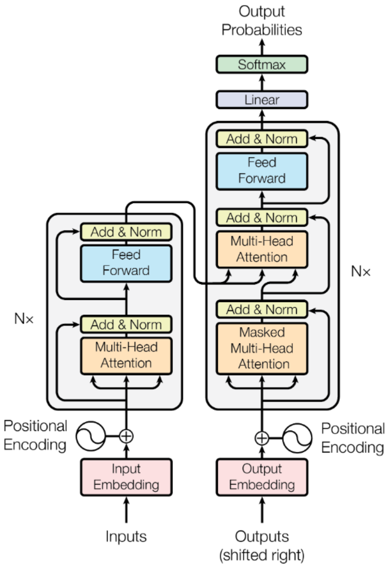
attention model

- 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/

Transformer encoder-decoder [Vaswani et al., 2017]



attention is all you need

- acausal self-attention in encoder
- causal self-attention in decoder
- cross-attention between encoder and decoder

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

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

Decoding

exact search

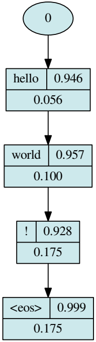
- generate every possible sentence T in target language
- compute score $p(T|S)$ for each
- pick best one

- 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 $\operatorname{argmax}_y p(y_i|S, y_{<i})$
- continue until we generate *<eos>*



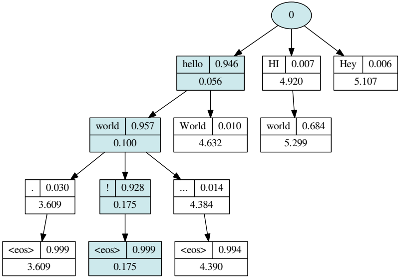
- efficient, but suboptimal

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

Advanced NMT [Sennrich et al., 2016a]

In order to achieve high quality NMT benefits from specific techniques. For instance:

- Subword models to allow translation of rare/unknown words
  $\rightarrow$ since networks have small, fixed vocabulary
- Back-translated monolingual data as additional training data
  $\rightarrow$ allows us to make use of extensive monolingual resources
- Dropout
  $\rightarrow$ Improves generalisation performance with small training data
- Virtual mini-batching
  $\rightarrow$ Improves generalization by tuning gradient noise

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

Byte pair encoding for word segmentation [Sennrich et al., 2016c]

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 [Sennrich et al., 2016c]

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 [Sennrich et al., 2016c]

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 **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 [Sennrich et al., 2016c]

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 [Sennrich et al., 2016c]

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
'**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 [Sennrich et al., 2016c]

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
'**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
  $\rightarrow$ trade-off between text length and vocabulary size

'**lo** 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

’**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

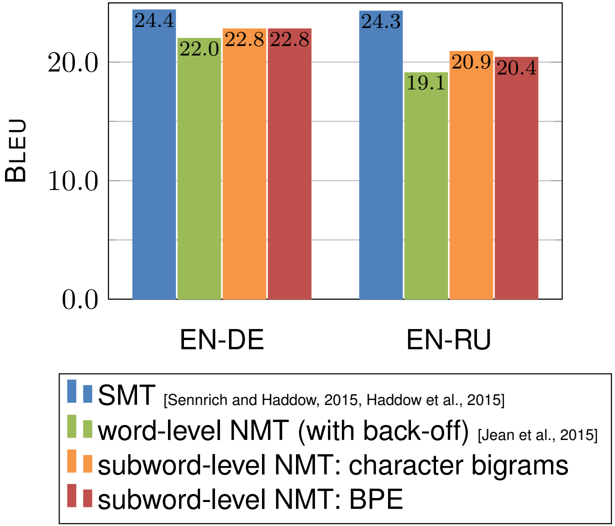
data

- WMT 15 English→German and English→Russian

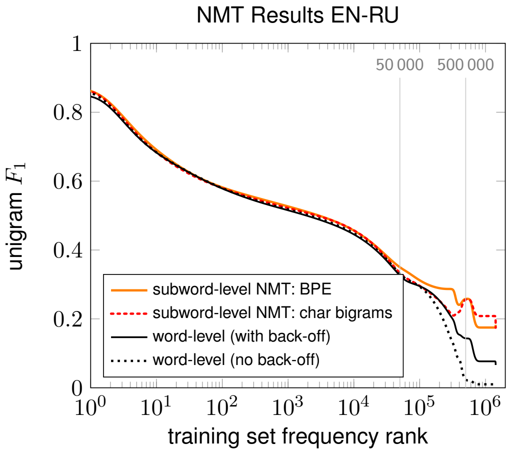
model

- 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 practice

- Use **Joint** BPE for same script languages
  - Just concatenate source and target, then train
  - Named-entities are split consistently

BPE in practice

- Use **Joint** BPE for same script languages
  - Just concatenate source and target, then train
  - Named-entities are split consistently
- merge operations: 30,000 - 80,000

BPE in practice

- Use **Joint** BPE for same script languages
  - Just concatenate source and target, then train
  - Named-entities are split consistently
- merge operations: 30,000 - 80,000
- for low resource, frequency threshold: 10 [Sennrich and Zhang, 2019]

BPE in practice

- Use **Joint** BPE for same script languages
  - Just concatenate source and target, then train
  - Named-entities are split consistently
- merge operations: 30,000 - 80,000
- for low resource, frequency threshold: 10 [Sennrich and Zhang, 2019]
- Transliterate when scripts are different
- E.g. 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

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

Monolingual Data in NMT

Why Monolingual Data for NMT?

- more training data
- more appropriate training data (domain adaptation)

Monolingual Data in NMT

encoder-decoder already conditions on previous target words

$\Downarrow$

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

Solution: Backtranslation [Sennrich et al., 2016b]

1. train a system in the **reverse** direction (Tgt$\rightarrow$Src)
2. translate target-language data to create a syntetic source Src'
3. flip the direction of the syntetic parallel corpus: Src'$\rightarrow$Tgt
4. merge with the true parallel data and train a Src$\rightarrow$Tgt system

Monolingual Training Instances

Backtranslation

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

Monolingual Training Instances

Backtranslation

- 1-1 mix of parallel and monolingual training instances
  - oversample parallel data if needed
- randomly sample from back-translated data
- training does not distinguish between real and synthetic parallel data
  - actually, it's better if it does [Caswell et al., 2019]

Why is monolingual data helpful?

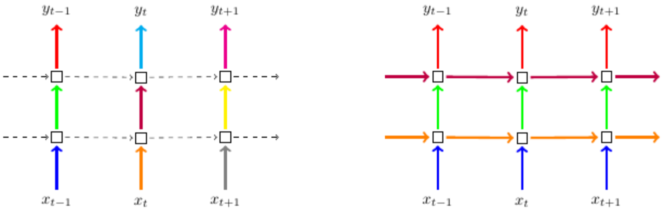
- Domain adaptation effect
- Reduces over-fitting
- Improves fluency

Why is monolingual data helpful?

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

- Additional techniques: copied monolingual data [Currey et al., 2017]
    - improves named entities accuracy

Dropout



[Gal, 2015]

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

Multi-lingual NMT

Why multilinguality?

- NMT models are usually trained on language pairs
- If we have N languages this implies N^2 models
  - poor scaling

Multi-lingual NMT

Why multilinguality?

- NMT models are usually trained on language pairs
- If we have N languages this implies N^2 models
  - poor scaling
- what if we have little or no parallel data for some pair?
  - e.g. little Cs$\leftrightarrow$Zh data, but plenty of Cs$\leftrightarrow$En and En$\leftrightarrow$Zh

Multi-lingual NMT

Why multilinguality?

- NMT models are usually trained on language pairs
- If we have N languages this implies N^2 models
    - poor scaling
- what if we have little or no parallel data for some pair?
    - e.g. little Cs$\leftrightarrow$Zh data, but plenty of Cs$\leftrightarrow$En and En$\leftrightarrow$Zh

Multi-lingual translation

- single model for multiple languages pairs
- we'd like to transfer training information between pairs
    - ideally, zero-shot translation

Multi-lingual NMT

Multi-lingual NMT techniques

- universal models
- direct pivoting
- backtranslation pivoting

Universal models [Ha et al., 2016]

model trained on multiple language pairs

- NMT models easily support multiple source languages
  - e.g. we want De→En and Fr→En
  - just mix the training corpora

Universal models [Ha et al., 2016]

model trained on multiple language pairs

- NMT models easily support multiple source languages
  - e.g. we want De→En and Fr→En
  - just mix the training corpora
- for multiple target languages append a tag
  - on the source side [Ha et al., 2016, Johnson et al., 2017]
  - or on the target side with forced decoding [Firat et al., 2016]

Universal models [Ha et al., 2016]

model trained on multiple language pairs

- NMT models easily support multiple source languages
  - e.g. we want De→En and Fr→En
  - just mix the training corpora
- for multiple target languages append a tag
  - on the source side [Ha et al., 2016, Johnson et al., 2017]
  - or on the target side with forced decoding [Firat et al., 2016]

pros & cons

- pro: works well on related languages
- pro: can be finetuned on parallel data
- con: inefficient for distant languages and/or different scripts
  - transliteration might help
- con: training set balancing issues

Direct pivoting

concatenate two models

- often one language (e.g. En) is strongly over-represented in the training data
- use it as a **pivot** language
    - e.g. we want Cs$\rightarrow$Zh
    - train Cs$\rightarrow$En and En$\rightarrow$Zh and concatenate them

Direct pivoting

concatenate two models

- often one language (e.g. En) is strongly over-represented in the training data
- use it as a **pivot** language
  - e.g. we want Cs$\rightarrow$Zh
  - train Cs$\rightarrow$En and En$\rightarrow$Zh and concatenate them

pros & cons

- pro: models can be optimized separately
- pro: allows language-specific pre- and post-processing
- pro: no negative interference between distant languages pairs
- con: can't use parallel data
- con: final system is more cumbersome

Backtranslation pivoting [Bawden et al., 2019]

pivot during training using backtraslations

- Example
  - we want En→Gu
  - we have little En↔Gu data, but plenty of En↔Hi and Hi↔Gu

1. train Hi→En
2. translate the Hi side of the the Hi↔Gu corpus to synthetic En'
3. pair back the original Gu to En' and flip it around to obtain En'↔Gu
4. merge with the true parallel data and train En→Gu

Backtranslation pivoting [Bawden et al., 2019]

pivot during training using backtraslations

- Example
  - we want En→Gu
  - we have little En↔Gu data, but plenty of En↔Hi and Hi↔Gu

1. train Hi→En
2. translate the Hi side of the the Hi↔Gu corpus to synthetic En'
3. pair back the original Gu to En' and flip it around to obtain En'↔Gu
4. merge with the true parallel data and train En→Gu

pros & cons

- pro: can use parallel and monolingual data
- pro: no negative interference between distant languages pairs
- pro: simple final system
- con: more complicated training

Backtranslation pivoting [Bawden et al., 2019]

pivot during training using backtraslations

- Example
  - we want En→Gu
  - we have little En↔Gu data, but plenty of En↔Hi and Hi↔Gu

1. train Hi→En
2. translate the Hi side of the the Hi↔Gu corpus to synthetic En'
3. pair back the original Gu to En' and flip it around to obtain En'↔Gu
4. merge with the true parallel data and train En→Gu

WMT 2019

- we used this setup for the Edinburgh's submission to WMT 2019
- + transliteration of Hi to Gu and semi-supervised training
- human evaluation results
  - En→Gu: first place
  - Gu→En: second place

Getting Started: Do it Yourself

- sample files and instructions for training NMT model
  `https://github.com/EdinburghNLP/wmt17-scripts`
  `https://github.com/EdinburghNLP/wmt17-transformer-scripts`
- pre-trained models to test decoding (and for further experiments)
  `http://statmt.org/rsennrich/wmt16_systems/`

(A small selection of) Resources

NMT tools

- Nematus (TensorFlow) `https://github.com/EdinburghNLP/nematus`
- Marian (C++/CUDA) `https://github.com/marian-nmt/marian-dev`
- Tensor2Tensor (TensorFlow) `https://github.com/tensorflow/tensor2tensor`
- Fairseq (PyTorch) `https://github.com/pytorch/fairseq`
- XLM (PyTorch) `https://github.com/facebookresearch/XLM`
- ...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
- The Illustrated Transformer http://jalammar.github.io/illustrated-transformer/

Acknowledgments

These slides have been adapted from a previous tutorial by **Rico Sennrich** and **Barry Haddow**

Questions

Thank you!

Bibliography I

Bai, S., Kolter, J. Z., and Koltun, V. (2018).
An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.
arXiv preprint arXiv:1803.01271.

Bawden, R., Bogoychev, N., Germann, U., Grundkiewicz, R., Kirefu, F., Miceli Barone, A. V., and Birch, A. (2019).
The university of edinburgh's submissions to the wmt19 news translation task.
In Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1), pages 103–115, Florence, Italy. Association for Computational Linguistics.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003).
A Neural Probabilistic Language Model.
J. Mach. Learn. Res., 3:1137–1155.

Caswell, I., Chelba, C., and Grangier, D. (2019).
Tagged back-translation.
In Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers), pages 53–63, Florence, Italy. Association for Computational Linguistics.

Cho, K. (2015).
Natural Language Understanding with Distributed Representation.
CoRR, abs/1511.07916.

Currey, A., Barone, A. V. M., and Heafield, K. (2017).
Copied monolingual data improves low-resource neural machine translation.
In Proceedings of the Second Conference on Machine Translation, pages 148–156.

Dai, Z., Yang, Z., Yang, Y., Cohen, W. W., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019).
Transformer-xl: Attentive language models beyond a fixed-length context.
arXiv preprint arXiv:1901.02860.

Bibliography II

Firat, O., Cho, K., and Bengio, Y. (2016).
Multi-way, multilingual neural machine translation with a shared attention mechanism.
arXiv preprint arXiv:1601.01073.

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., and Waibel, A. (2016).
Toward multilingual neural machine translation with universal encoder and decoder.
arXiv preprint arXiv:1611.04798.

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.

Bibliography III

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., et al. (2017).
Google's multilingual neural machine translation system: Enabling zero-shot translation.
Transactions of the Association for Computational Linguistics, 5:339–351.

Miceli Barone, A. V., Helcl, J., Sennrich, R., Haddow, B., and Birch, A. (2017).
Deep architectures for neural machine translation.
In Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017, pages 99–107.

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.

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.

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.

Bibliography IV

Sennrich, R., Haddow, B., and Birch, A. (2016a).
Edinburgh Neural Machine Translation Systems for WMT 16.
In Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 368–373, Berlin, Germany. Association for Computational Linguistics.

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

Sennrich, R., Haddow, B., and Birch, A. (2016c).
Neural Machine Translation of Rare Words with Subword Units.
In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R. and Zhang, B. (2019).
Revisiting low-resource neural machine translation: A case study.
arXiv preprint arXiv:1905.11901.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017).
Attention is all you need.
In Advances in neural information processing systems, pages 5998–6008.

Wu, F., Fan, A., Baevski, A., Dauphin, Y., and Auli, M. (2019).
Pay less attention with lightweight and dynamic convolutions.
In International Conference on Learning Representations.