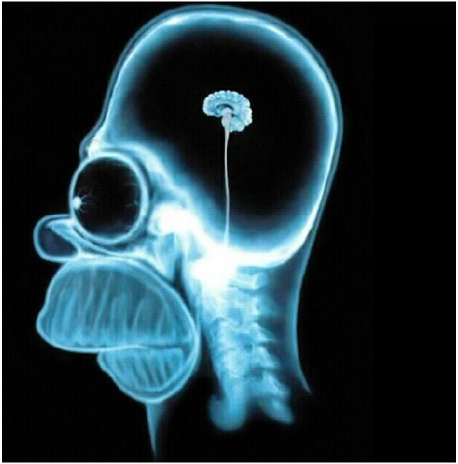
Introduction to Neural Machine Translation

Fabienne Cap

UPPSALA
UNIVERSITY
SWEDEN

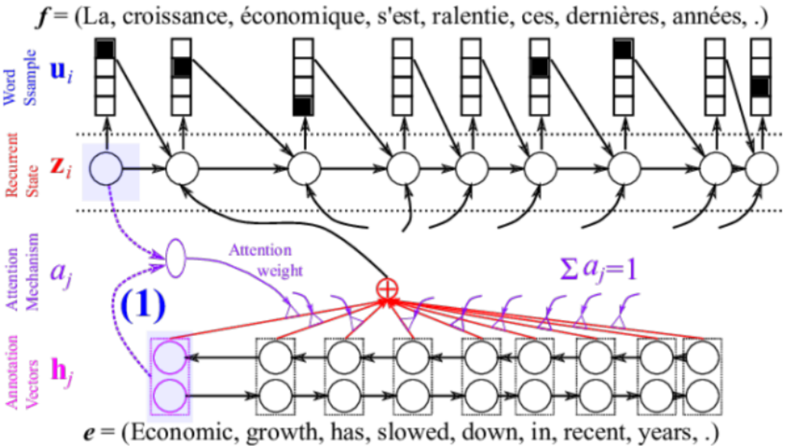
What is Neural Machine Translation?



"Neural Machine Translation is the approach of modeling the entire MT process via one big artificial neural network." (Manning, 2016)

NMT Example

… repeated from Introduction Lecture:



Taken from (Baldahau et al. 2014)

NMT Timeline

1987	**Early encoder-decoder, with vocabulary size 30-40** (Allen, 1987)
...	
2013	**Pure neural MT system presented** (Kalchbrenner & Blunsom, 2013)
2014	**Competitive encoder-decoder for large-scale MT** (Bahdanau et al., 2015, Luong et al., 2014)
2015	**NMT systems in shared tasks** performs well in WMT, state-of-the-art at IWSLT
2016	**NMT systems top most language pairs in WMT**
2016	**Commercial deployments of NMT launched**

Taken from EACL 2017 Tutorial on Practical NMT

Overview

NMT vs. SMT

A typical NMT system

Shortcomings of NMT and proposed solutions

Outlook

Overview

NMT vs. SMT

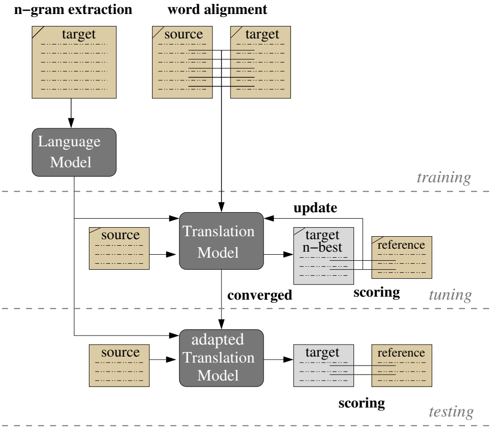
A typical NMT system

Shortcomings of NMT and proposed solutions

Outlook

NMT vs. SMT: Architecture

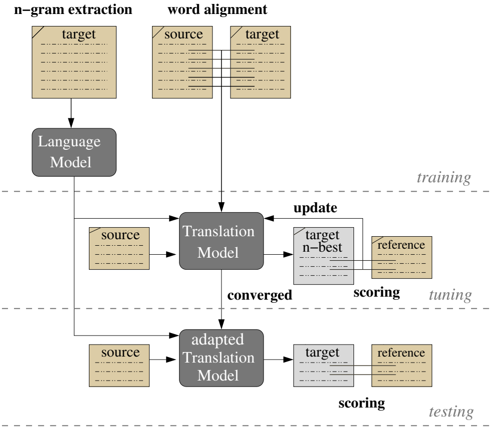
Review: SMT Architecture



- many components → pipeline architecture
- context-independent translations
- idependent models (TM, LM, RM)
- learn feature weights using minimum error rate training

NMT vs. SMT: Architecture

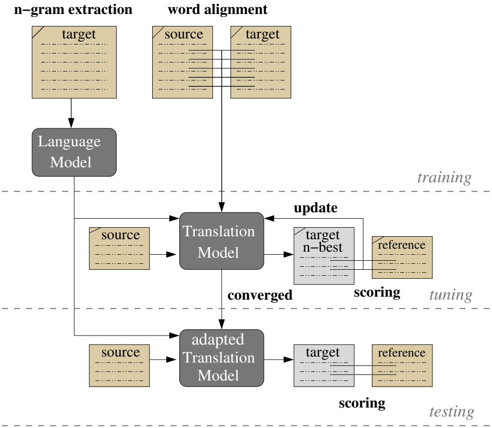
Review: SMT Architecture



- many components → pipeline architecture
- context-independent translations
- idependent models (TM, LM, RM)
- learn feature weights using minimum error rate training

NMT vs. SMT: Architecture

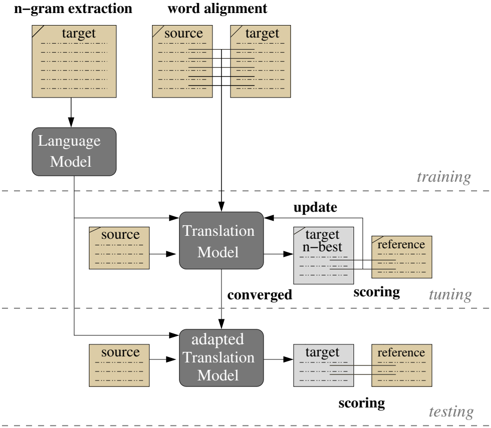
Review: SMT Architecture



- many components → pipeline architecture
- context-independent translations
- idependent models (TM, LM, RM)
- learn feature weights using minimum error rate training

NMT vs. SMT: Architecture

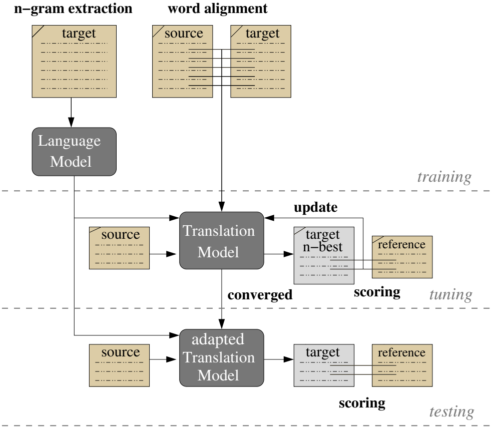
Review: SMT Architecture



- many components $\rightarrow$ pipeline architecture
- context-independent translations
- idependent models (TM, LM, RM)
- learn feature weights using minimum error rate training

NMT vs. SMT: Architecture

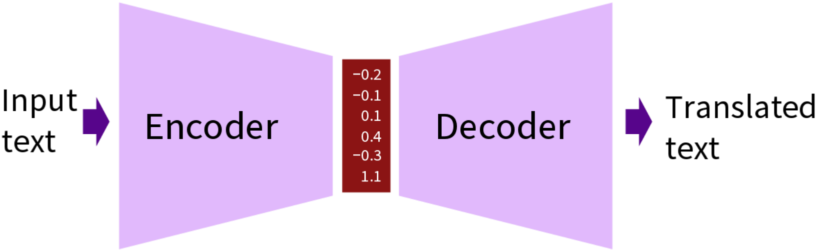
Review: SMT Architecture



- many components $\rightarrow$ pipeline architecture
- context-independent translations
- idependent models (TM, LM, RM)
- learn feature weights using minimum error rate training

NMT vs. SMT: Architecture

A typical **neural** encoder-decoder architecture

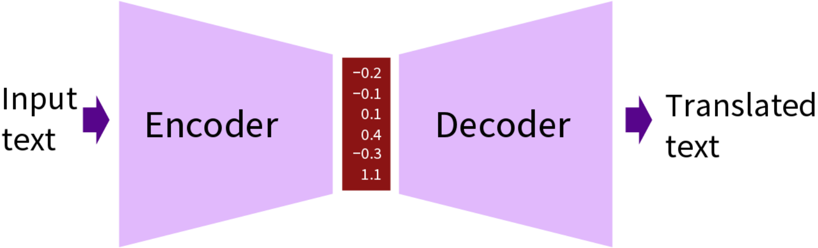


- everything in one large model
- all parameters optimised globally
- no explicit division into TM, LM RM

Based on ACL NMT Tutorial 2016

NMT vs. SMT: Architecture

A typical **neural** encoder-decoder architecture

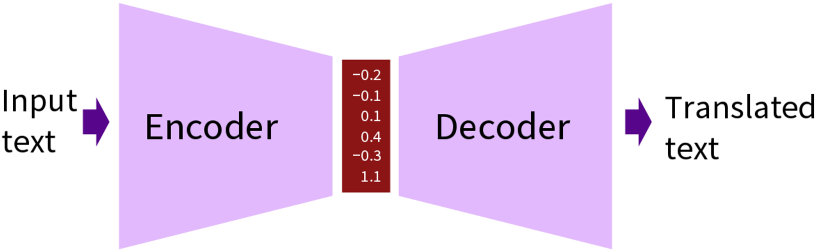


- everything in one large model
- all parameters optimised globally
- no explicit division into TM, LM RM

Based on ACL NMT Tutorial 2016

NMT vs. SMT: Architecture

A typical **neural** encoder-decoder architecture

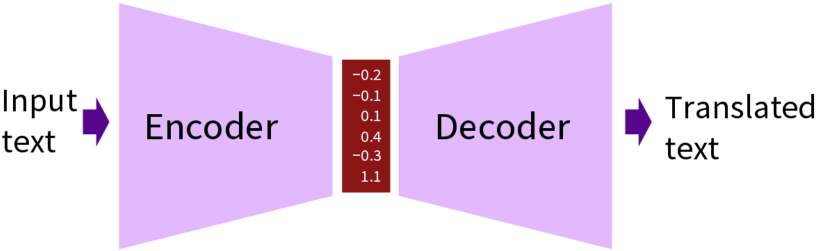


- everything in one large model
- all parameters optimised globally
- no explicit division into TM, LM RM

Based on ACL NMT Tutorial 2016

NMT vs. SMT: Architecture

A typical **neural** encoder-decoder architecture



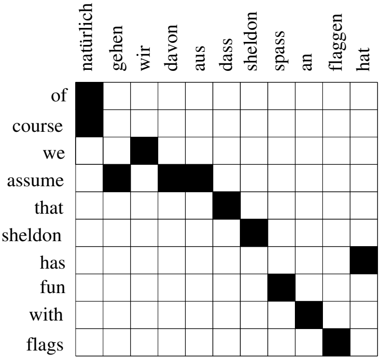
- everything in one large model
- all parameters optimised globally
- no explicit division into TM, LM RM

Based on ACL NMT Tutorial 2016

SMT vs. NMT: Word Alignment

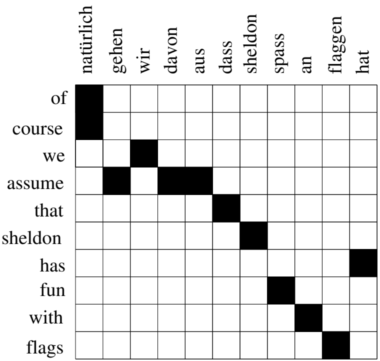
SMT vs. NMT: Word Alignment

SMT: Word Alignments

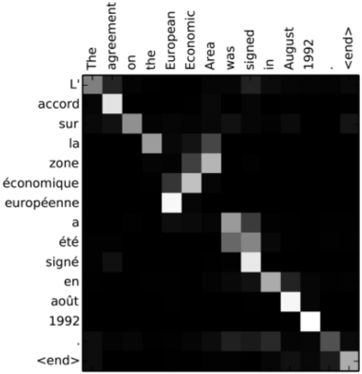


SMT vs. NMT: Word Alignment

SMT: Word Alignments



NMT: Attention Models



Taken from (Bahdanau et al., 2015)

SMT vs. NMT: Context

SMT:

- phrase table weights gave a **context-independent** translation score
- use language models (LM) to ensure **target language** fluency

Based on ACL NMT Tutorial 2016

SMT vs. NMT: Context

SMT:

- phrase table weights gave a **context-independent** translation score
- use language models (LM) to ensure **target language** fluency

NMT:

- all translations are **context-dependent**!
- generate a translation **with an LM** also conditioned on the **source language**

Based on ACL NMT Tutorial 2016

NMT vs. SMT: Performance

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

Taken from EACL 2017 Tutorial on Practical NMT

Overview

NMT vs. SMT

A typical NMT system

Shortcomings of NMT and proposed solutions

Outlook

Overview

NMT vs. SMT

A typical NMT system

Shortcomings of NMT and proposed solutions

Outlook

A typical NMT system

- NMT is nothing but a particular variety of SMT:
  $P(e|f)$ is learned using neural networks

- The translation is formulated as a prediction problem, similar to a language model but trained on both source and target sequences

- **Encoder-decoder models**:
  use a recurrent neural network (RNN) to read a source language sequence and predict a target language sequence

Based on https://github.com/neubig/nmt-tips and ACL 2016 Tutorial on NMT

A typical NMT system

- NMT is nothing but a particular variety of SMT:
  P(e|f) is learned using neural networks
- The translation is formulated as a prediction problem, similar to a language model but trained on both source and target sequences
- **Encoder-decoder models**:
  use a recurrent neural network (RNN) to read a source language sequence and predict a target language sequence

Based on https://github.com/neubig/nmt-tips and ACL 2016 Tutorial on NMT

A typical NMT system

- NMT is nothing but a particular variety of SMT:
  $P(e|f)$ is learned using neural networks

- The translation is formulated as a prediction problem, similar to a language model but trained on both source and target sequences

- **Encoder-decoder models**:
  use a recurrent neural network (RNN) to read a source language sequence and predict a target language sequence

Based on https://github.com/neubig/nmt-tips and ACL 2016 Tutorial on NMT

A typcial NMT system: Encoder-Decoder Model

1) **Encode** source sentence: convert into fixed-length vector
   $source\_emb_j = WORDREP(source\_word_j, parameters)$
2) Map to **hidden state** using an RNN:
   $hidden_j = RNN(h_{j-1}, source\_emb_j, parameters)$
3) **Decode:**

Based on https://github.com/neubig/nmt-tips

A typcial NMT system: Encoder-Decoder Model

1) **Encode** source sentence: convert into fixed-length vector
   $source\_emb_j = WORDREP(source\_word_j, parameters)$
2) Map to **hidden state** using an RNN:
   $hidden_j = RNN(h_{j-1}, source\_emb_j, parameters)$
3) **Decode:**

Based on https://github.com/neubig/nmt-tips

A typcial NMT system: Encoder-Decoder Model

1) **Encode** source sentence: convert into fixed-length vector
$source\_emb_j = WORDREP(source\_word_j, parameters)$
2) Map to **hidden state** using an RNN:
$hidden_j = RNN(h_{j-1}, source\_emb_j, parameters)$
3) **Decode:**

Based on https://github.com/neubig/nmt-tips

A typcial NMT system: Encoder-Decoder Model

1) **Encode** source sentence: convert into fixed-length vector
   $source\_emb_j = WORDREP(source\_word_j, parameters)$
2) Map to **hidden state** using an RNN:
   $hidden_j = RNN(h_{j-1}, source\_emb_j, parameters)$
3) **Decode:**

Based on https://github.com/neubig/nmt-tips

A typcial NMT system: Encoder-Decoder Model

1) **Encode** source sentence: convert into fixed-length vector
   $source\_emb_j = WORDREP(source\_word_j, parameters)$
2) Map to **hidden state** using an RNN:
   $hidden_j = RNN(h_{j-1}, source\_emb_j, parameters)$
3) **Decode:**
   a) first hidden state of decoder = last hidden state of encoder

Based on https://github.com/neubig/nmt-tips

A typcial NMT system: Encoder-Decoder Model

1) **Encode** source sentence: convert into fixed-length vector
$source\_emb_j = WORDREP(source\_word_j, parameters)$

2) Map to **hidden state** using an RNN:
$hidden_j = RNN(h_{j-1}, source\_emb_j, parameters)$

3) **Decode:**
a) first hidden state of decoder = last hidden state of encoder
b) predict target word probabilities:
$probability\_estim = SOFTMAX(g_{i-1}; parameters)$

Based on https://github.com/neubig/nmt-tips

A typcial NMT system: Encoder-Decoder Model

1) **Encode** source sentence: convert into fixed-length vector
$source\_emb_j = WORDREP(source\_word_j, parameters)$

2) Map to **hidden state** using an RNN:
$hidden_j = RNN(h_{j-1}, source\_emb_j, parameters)$

3) **Decode:**
a) first hidden state of decoder = last hidden state of encoder
b) predict target word probabilities:
$probability\_estim = SOFTMAX(g_{i-1}; parameters)$
c) pick most probable word:
$target\_word_i = ARGMAX_k(probability\_estim_i[k])$

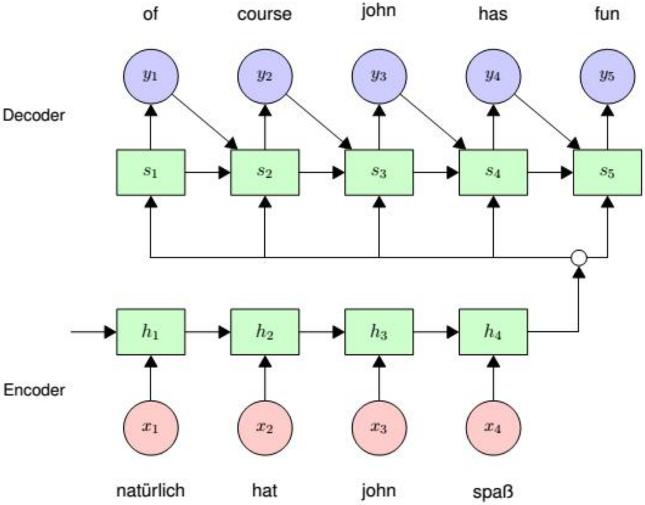
Based on https://github.com/neubig/nmt-tips

A typcial NMT system: Encoder-Decoder Model

1) **Encode** source sentence: convert into fixed-length vector
$source\_emb_j = WORDREP(source\_word_j, parameters)$
2) Map to **hidden state** using an RNN:
$hidden_j = RNN(h_{j-1}, source\_emb_j, parameters)$
3) **Decode:**
a) first hidden state of decoder = last hidden state of encoder
b) predict target word probabilities:
$probability\_estim = SOFTMAX(g_{i-1}; parameters)$
c) pick most probable word:
$target\_word_i = ARGMAX_k(probability\_estim_i[k])$
d) update the hidden state with this predicted value:
$target\_emb_i = WORDREP(target\_word_i, parameters)$
$g_i = RNN(g_{i-1}, target\_emb_{i-1}, parameters)$

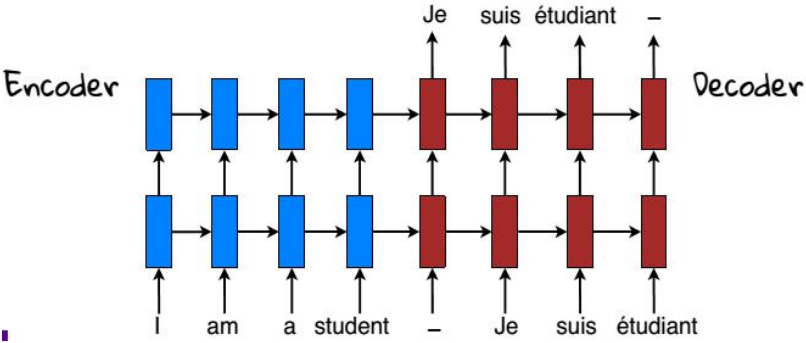
Based on https://github.com/neubig/nmt-tips

A typical NMT system: Encoder-Decoder Model



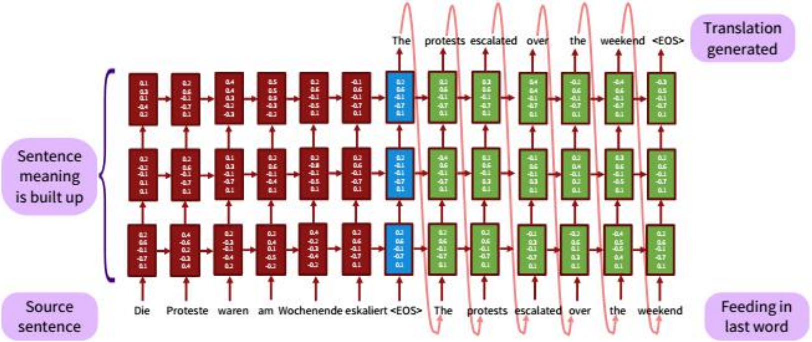
Taken from the EACL 2017 Tutorial on Practical NMT

A typical NMT system: Encoder-Decoder Model



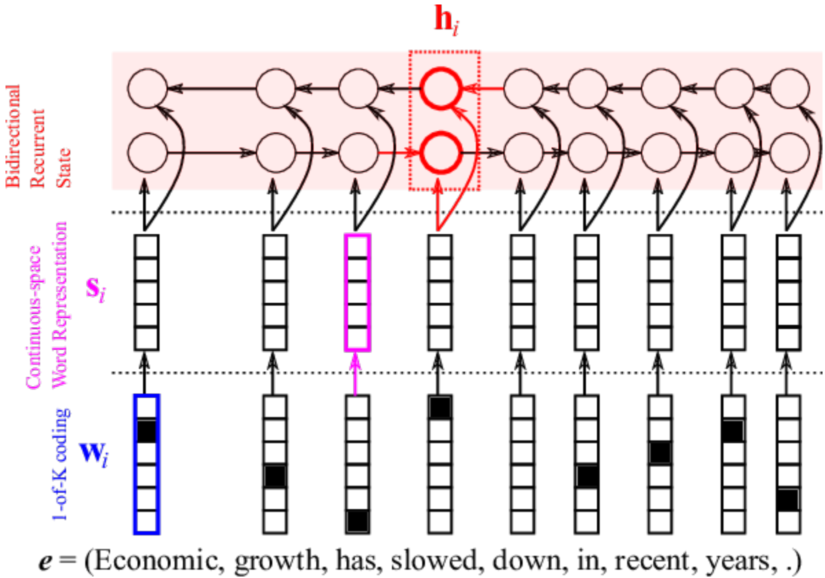
Taken from the ACL 2016 Tutorial on NMT

A typical NMT system: Encoder-Decoder Model

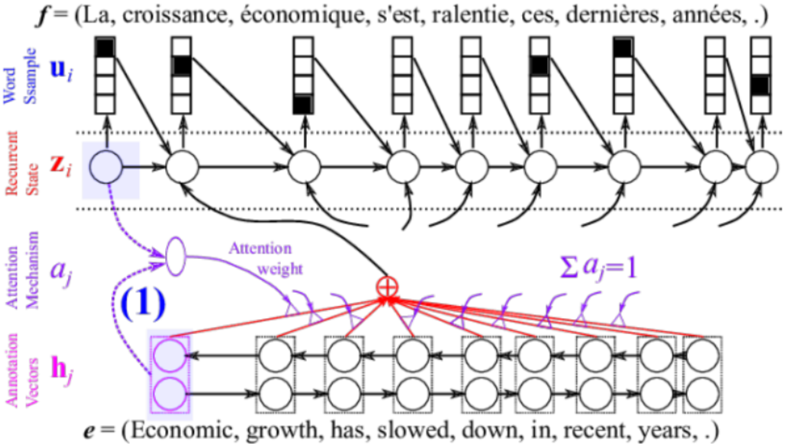


Taken from the ACL 2016 Tutorial on NMT

A typical NMT system: Encoder-Decoder Model



A typical NMT system: Encoder-Decoder Model



Taken from (Baldahau et al. 2014)

A typical NMT system: Encoder-Decoder Model

1) **Encode** source sentence: convert into fixed-length vector
$source\_emb_j = WORDREP(source\_word_j, parameters)$

2) Map to **hidden state** using an RNN:
$hidden_j = RNN(h_{j-1}, source\_emb_j, parameters)$

3) **Decode:**
a) first hidden state of decoder = last hidden state of encoder
b) predict target word probabilities:
$probability\_estim = SOFTMAX(g_{i-1}; parameters)$
c) pick most probable word:
$target\_word_i = ARGMAX_k(probability\_estim_i[k])$
d) update the hidden state with this predicted value:
$target\_emb_i = WORDREP(target\_word_i, parameters)$
$g_i = RNN(g_{i-1}, target\_emb_{i-1}, parameters)$

Based on https://github.com/neubig/nmt-tips

A typical NMT system: Encoder-Decoder Model

1) **Encode** source sentence: convert into fixed-length vector
$source\_emb_j = WORDREP(source\_word_j, parameters)$

2) Map to **hidden state** using an RNN:
$hidden_j = RNN(h_{j-1}, source\_emb_j, parameters)$

3) **Decode:**

a) first hidden state of decoder = last hidden state of encoder

b) predict target word probabilities:
$probability\_estim = SOFTMAX(g_{i-1}; parameters)$

c) pick most probable word:
$target\_word_i = ARGMAX_k(probability\_estim_i[k])$

d) update the hidden state with this predicted value:
$target\_emb_i = WORDREP(target\_word_i, parameters)$
$g_i = RNN(g_{i-1}, target\_emb_{i-1}, parameters)$

This is a training loop! Based on https://github.com/neubig/nmt-tips

Training NMT Models with Maximum Likelihood

- the parameters (ϕ) of the models need to be learned
- standard way of doing this: maximise the log lilkelihood of the training data:
  $\phi' = ARGMAX_{\phi}(\sum_{E,F} logP(E|F;\phi))$
- equivalent: minimise the negative log likelihood:
  $\phi' = ARGMIN_{\phi}(-\sum_{E,F} logP(E|F;\phi)$
- minimisation using stochastic gradient descent (SGD): calculate gradient of the negative log probability:
  $\nabla\phi - logP(E|F;\phi)$
- then update the parameters based on an update rule:
  $\phi \leftarrow UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi)$
- substract the gradient of the negative log likelihood multiplied by a learning rate γ
  $SGD\_UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi), \gamma) :=$
  $\phi - \gamma * \nabla_{\phi} - logP(E|F;\phi)$

Based on https://github.com/neubig/nmt-tips

Training NMT Models with Maximum Likelihood

- the parameters (ϕ) of the models need to be learned
- standard way of doing this: maximise the log lilkelihood of the training data:
  $\phi' = ARGMAX_{\phi}(\sum_{E,F} logP(E|F;\phi))$
- equivalent: minimise the negative log likelihood:
  $\phi' = ARGMIN_{\phi}(-\sum_{E,F} logP(E|F;\phi)$
- minimisation using stochastic gradient descent (SGD): calculate gradient of the negative log probability:
  $\nabla\phi - logP(E|F;\phi)$
- then update the parameters based on an update rule:
  $\phi \leftarrow UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi)$
- substract the gradient of the negative log likelihood multiplied by a learning rate γ
  $SGD\_UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi), \gamma) :=$
  $\phi - \gamma * \nabla_{\phi} - logP(E|F;\phi)$

Based on https://github.com/neubig/nmt-tips

Training NMT Models with Maximum Likelihood

- the parameters (ϕ) of the models need to be learned
- standard way of doing this: maximise the log lilkelihood of the training data:
  $\phi' = ARGMAX_{\phi}(\sum_{E,F} logP(E|F;\phi))$
- equivalent: minimise the negative log likelihood:
  $\phi' = ARGMIN_{\phi}(-\sum_{E,F} logP(E|F;\phi)$
- minimisation using stochastic gradient descent (SGD): calculate gradient of the negative log probability:
  $\nabla\phi - logP(E|F;\phi)$
- then update the parameters based on an update rule:
  $\phi \leftarrow UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi)$
- substract the gradient of the negative log likelihood multiplied by a learning rate γ
  $SGD\_UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi), \gamma) :=$
  $\phi - \gamma * \nabla_{\phi} - logP(E|F;\phi)$

Based on https://github.com/neubig/nmt-tips

Training NMT Models with Maximum Likelihood

- the parameters (ϕ) of the models need to be learned
- standard way of doing this: maximise the log lilkelihood of the training data:
  $\phi' = ARGMAX_{\phi}(\sum_{E,F} logP(E|F;\phi))$
- equivalent: minimise the negative log likelihood:
  $\phi' = ARGMIN_{\phi}(-\sum_{E,F} logP(E|F;\phi)$
- minimisation using stochastic gradient descent (SGD): calculate gradient of the negative log probability:
  $\nabla\phi - logP(E|F;\phi)$
- then update the parameters based on an update rule:
  $\phi \leftarrow UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi)$
- substract the gradient of the negative log likelihood multiplied by a learning rate γ
  $SGD\_UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi), \gamma) :=$
  $\phi - \gamma * \nabla_{\phi} - logP(E|F;\phi)$

Based on https://github.com/neubig/nmt-tips

Training NMT Models with Maximum Likelihood

- the parameters (ϕ) of the models need to be learned
- standard way of doing this: maximise the log lilkelihood of the training data:
  $\phi' = ARGMAX_{\phi}(\sum_{E,F} logP(E|F;\phi))$
- equivalent: minimise the negative log likelihood:
  $\phi' = ARGMIN_{\phi}(-\sum_{E,F} logP(E|F;\phi)$
- minimisation using stochastic gradient descent (SGD): calculate gradient of the negative log probability:
  $\nabla\phi - logP(E|F;\phi)$
- then update the parameters based on an update rule:
  $\phi \leftarrow UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi)$
- substract the gradient of the negative log likelihood multiplied by a learning rate γ
  $SGD\_UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi), \gamma) :=$
  $\phi - \gamma * \nabla_{\phi} - logP(E|F;\phi)$

Based on https://github.com/neubig/nmt-tips

Training NMT Models with Maximum Likelihood

- the parameters (ϕ) of the models need to be learned
- standard way of doing this: maximise the log lilkelihood of the training data:
  $\phi' = ARGMAX_{\phi}(\sum_{E,F} logP(E|F;\phi))$
- equivalent: minimise the negative log likelihood:
  $\phi' = ARGMIN_{\phi}(-\sum_{E,F} logP(E|F;\phi)$
- minimisation using stochastic gradient descent (SGD): calculate gradient of the negative log probability:
  $\nabla\phi - logP(E|F;\phi)$
- then update the parameters based on an update rule:
  $\phi \leftarrow UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi)$
- substract the gradient of the negative log likelihood multiplied by a learning rate γ
  $SGD\_UPDATE(\phi, \nabla_{\phi} - logP(E|F;\phi), \gamma) :=$
  $\phi - \gamma * \nabla_{\phi} - logP(E|F;\phi)$

Based on https://github.com/neubig/nmt-tips

Excursion: Gradient Descent

General concept in machine learning:
minimise the distance between what **has been predicted** and what **should have been predicted**.

Excursion: Gradient Descent

General concept in machine learning:
minimise the distance between what **has been predicted** and what **should have been predicted**.

Andrew Ng's Coursera course on Machine Learning:
https://www.youtube.com/watch?v=LN0PLnDpGN4&t=121s

A typical NMT system: Mini-Batching

- **Problem:**
  Gradient descent for each single sentence takes time!
- **Solution:**
  Update the gradients for multiple sentences at the same time.
  This is called **Mini-batching**

Based on EACL 2017 Tutorial on Practical NMT,
https://github.com/neubig/nmt-tips

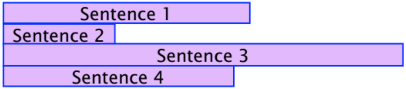
A typical NMT system: Mini-Batching

- **Problem:**
  Gradient descent for each single sentence takes time!
- **Solution:**
  Update the gradients for multiple sentences at the same time.
  This is called **Mini-batching**

Based on EACL 2017 Tutorial on Practical NMT,
https://github.com/neubig/nmt-tips

A typical NMT system: Mini-Batching

- **Problem:**
  Gradient descent for each single sentence takes time!
- **Solution:**
  Update the gradients for multiple sentences at the same time.
  This is called **Mini-batching**



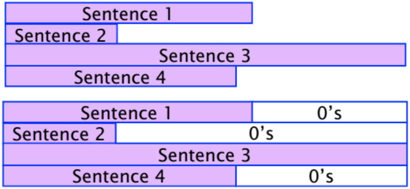
But: needs same vector length

Based on EACL 2017 Tutorial on Practical NMT,

https://github.com/neubig/nmt-tips

A typical NMT system: Mini-Batching

- **Problem:**
  Gradient descent for each single sentence takes time!
- **Solution:**
  Update the gradients for multiple sentences at the same time.
  This is called **Mini-batching**



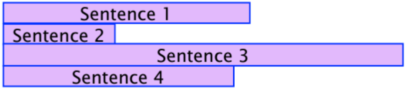
But: needs same vector length

padding, masking: fill with '0'
→ works! **But:** wasteful

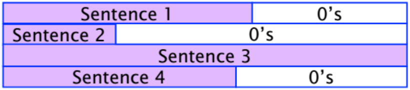
Based on EACL 2017 Tutorial on Practical NMT,
https://github.com/neubig/nmt-tips

A typical NMT system: Mini-Batching

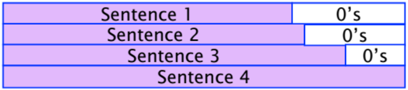
- **Problem:**
  Gradient descent for each single sentence takes time!
- **Solution:**
  Update the gradients for multiple sentences at the same time.
  This is called **Mini-batching**



But: needs same vector length



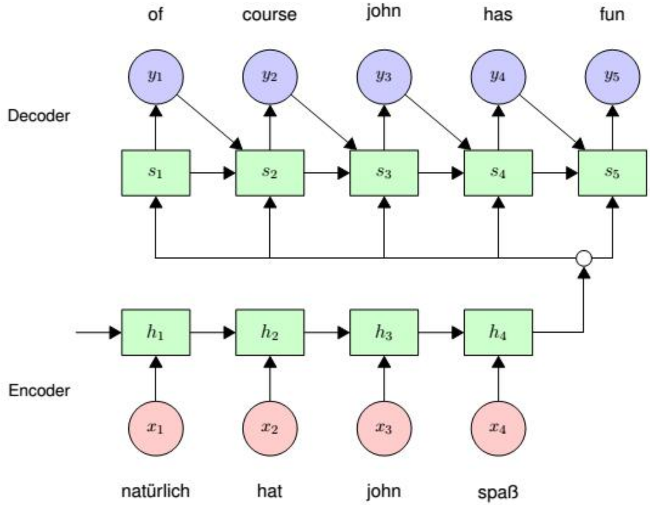
padding, masking: fill with '0'
$\rightarrow$ works! **But:** wasteful



Better:
sort wrt. to sentence length

Based on EACL 2017 Tutorial on Practical NMT,
https://github.com/neubig/nmt-tips

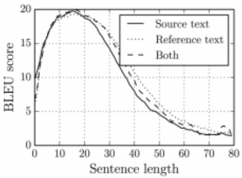
A typcial NMT system: Attention Mechanism



Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism

- Summary vector is an information bottleneck
- **Problem:** Sentence length! Fixed sized representation degrades as sentence lenght increases (Cho et al. 2014)

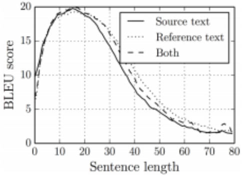


- Reversing source sequence brings some improvement (Sutskever et al. 2014)
- **Solution:** Attention
  - compute context vectors as weighted average of source hidden states
  - weights computed by feed-forward network with softmax activation

Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism

- Summary vector is an information bottleneck
- **Problem:** Sentence length! Fixed sized representation degrades as sentence lenght increases (Cho et al. 2014)

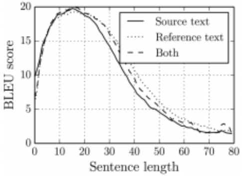


- Reversing source sequence brings some improvement (Sutskever et al. 2014)
- **Solution:** Attention

Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism

- Summary vector is an information bottleneck
- **Problem:** Sentence length! Fixed sized representation degrades as sentence lenght increases (Cho et al. 2014)

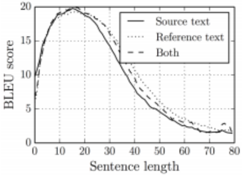


- Reversing source sequence brings some improvement (Sutskever et al. 2014)
- **Solution:** Attention
  - compute context vectors as weighted average of source hidden states
  - weights computed by feed-forward network with softmax activation

Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism

- Summary vector is an information bottleneck
- **Problem:** Sentence length! Fixed sized representation degrades as sentence lenght increases (Cho et al. 2014)

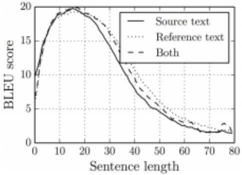


- Reversing source sequence brings some improvement (Sutskever et al. 2014)
- **Solution:** Attention
  - compute context vectors as weighted average of source hidden states
  - weights computed by feed-forward network with softmax activation

Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism

- Summary vector is an information bottleneck
- **Problem:** Sentence length! Fixed sized representation degrades as sentence lenght increases (Cho et al. 2014)

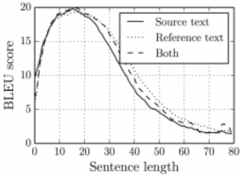


- Reversing source sequence brings some improvement (Sutskever et al. 2014)
- **Solution:** Attention
  - compute context vectors
    as weighted average of source hidden states
  - weights computed by feed-forward network
    with softmax activation

Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism

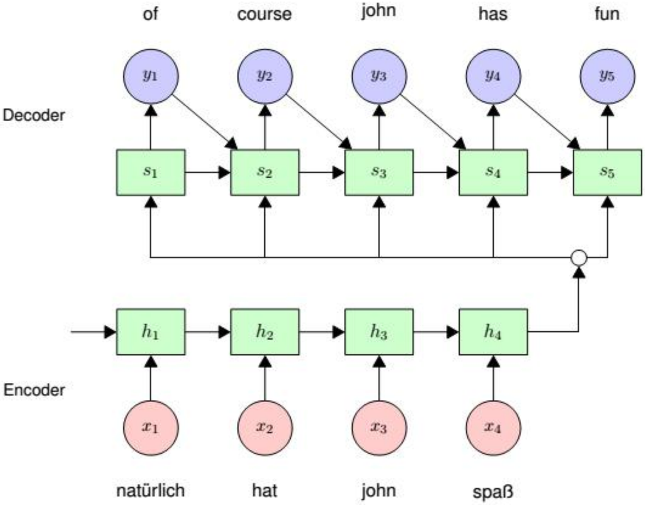
- Summary vector is an information bottleneck
- **Problem:** Sentence length! Fixed sized representation degrades as sentence lenght increases (Cho et al. 2014)



- Reversing source sequence brings some improvement (Sutskever et al. 2014)
- **Solution:** Attention
  - compute context vectors
    as weighted average of source hidden states
  - weights computed by feed-forward network
    with softmax activation

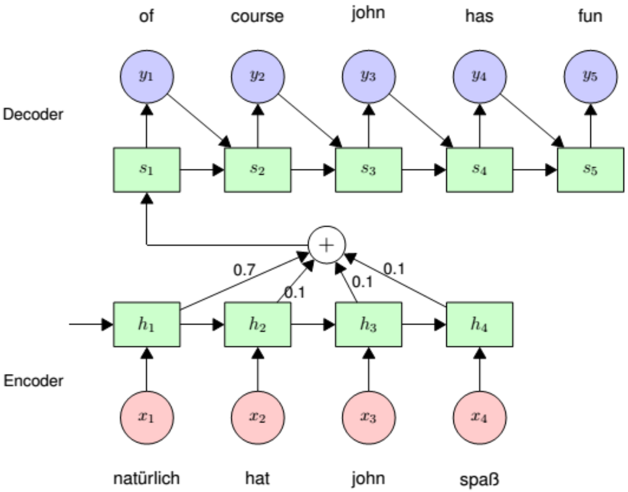
Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism



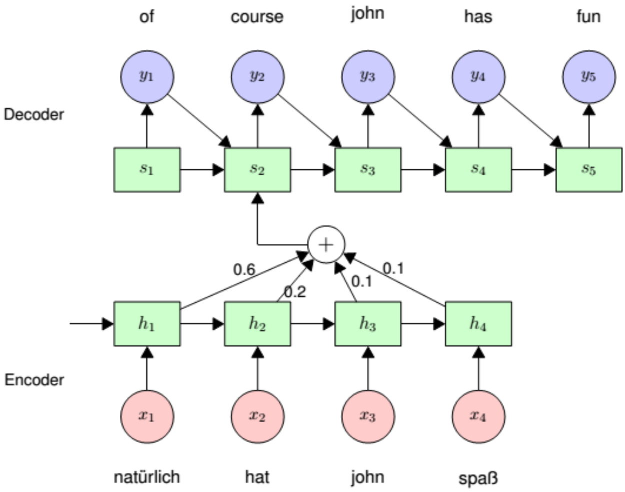
Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism



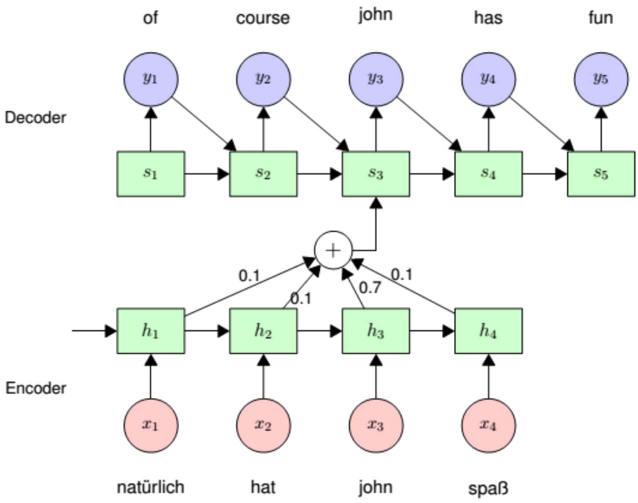
Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism



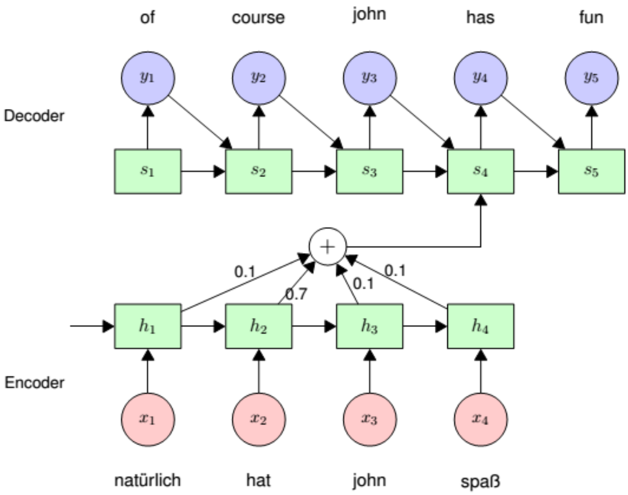
Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism



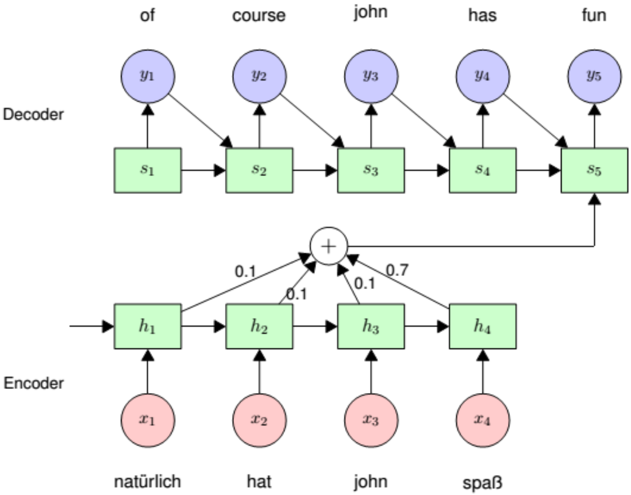
Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism



Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism



Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism

- side effect: we obtain alignments between the source and target sentence
- **but:** no guarantee that it corresponds to alignment! Information can also flow along recurrent connections.

Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism

- side effect: we obtain alignments between the source and target sentence
- **but:** no guarantee that it corresponds to alignment! Information can also flow along recurrent connections.

Taken from EACL 2017 Tutorial on Practical NMT

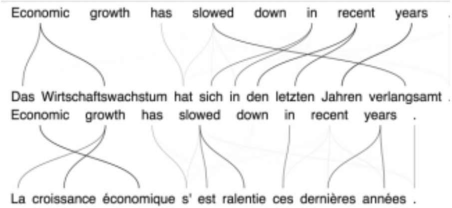
A typcial NMT system: Attention Mechanism

- side effect: we obtain alignments between the source and target sentence
- **but:** no guarantee that it corresponds to alignment! Information can also flow along recurrent connections.

Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism

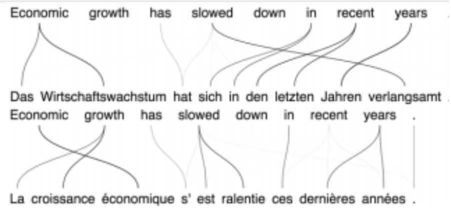
- side effect: we obtain alignments between the source and target sentence
- **but:** no guarantee that it corresponds to alignment! Information can also flow along recurrent connections.

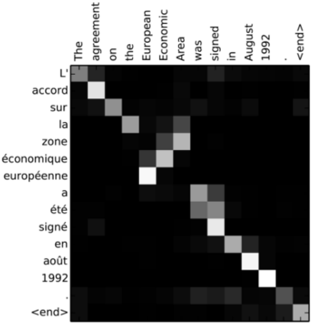


Taken from EACL 2017 Tutorial on Practical NMT

A typcial NMT system: Attention Mechanism

- side effect: we obtain alignments between the source and target sentence
- **but:** no guarantee that it corresponds to alignment! Information can also flow along recurrent connections.





Taken from EACL 2017 Tutorial on Practical NMT

Overview

NMT vs. SMT

A typical NMT system

Shortcomings of NMT and proposed solutions

Outlook

Overview

NMT vs. SMT

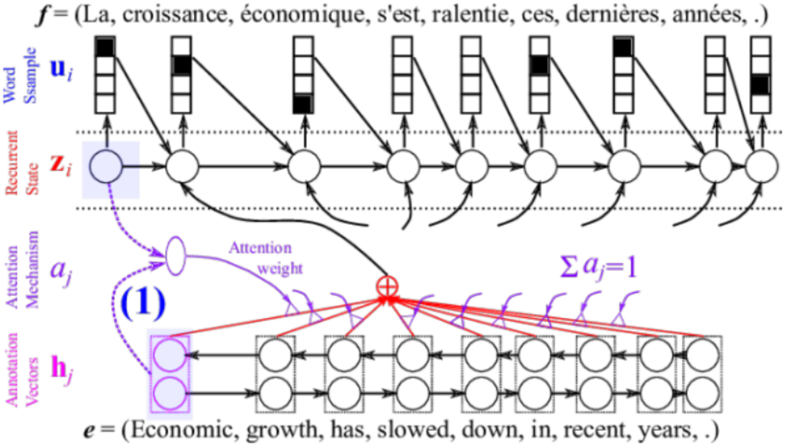
A typical NMT system

Shortcomings of NMT and proposed solutions

Outlook

NMT Example

... repeated from Introduction Lecture:



END OF TODAY

This is the end. Hold your breath and count to ten.