Machine Translation

Natural Language Processing (COM4513/6513)

Fred Blain

f.blain@sheffield.ac.uk / @fblain

Lecture 9 — May 7th, 2020

The University Of Sheffield.

Previously

- ↪ **Vector Representations of Text**
- ↪ **Language Modelling**
- ↪ **Sequence Labelling and Part-of-Speech Tagging**
- ↪ **Recurrent Networks and Neural Language Modelling**

Today

- Introduce a new task: **Machine Translation**

Today

- Introduce a new task: **Machine Translation**

- Introduce a new neural architecture: **Sequence-to-sequence**

Today

- Introduce a new task: **Machine Translation**
- Introduce a new neural architecture: **Sequence-to-sequence**
- Talk about Attention & **Self-Attention**

[...] When I look at an article in Russian, I say : "This is really written in English, but it has been coded in some strange symbols. I now proceed to decode." [...]

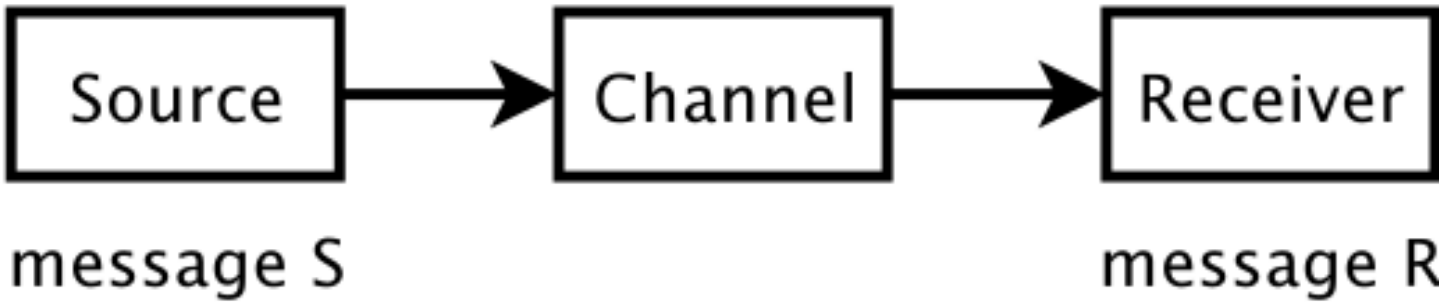
— W. Weaver

[...] When I look at an article in Russian, I say : "This is really written in English, but it has been coded in some strange symbols. I now proceed to decode." [...]

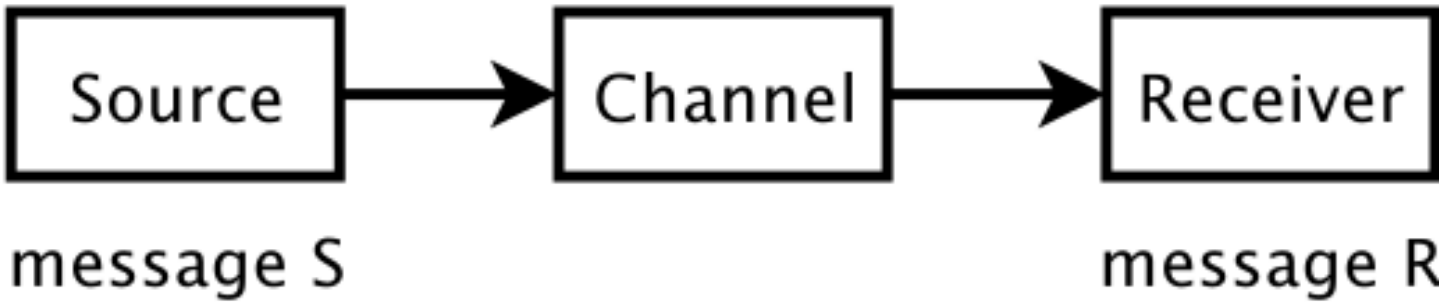
— W. Weaver (1947)

Letter to N. Wiener, suggesting that cryptanalysis techniques might be applied to translation, and that a computer could be built for the purpose.

Machine Translation



- we observe a distorted message R (e.g. foreign string **f**)
- we want to recover the original message S (e.g. English string **e**)



- we observe a distorted message R (e.g. foreign string **f**)
- we want to recover the original message S (e.g. English string **e**)

Noisy Channel Model

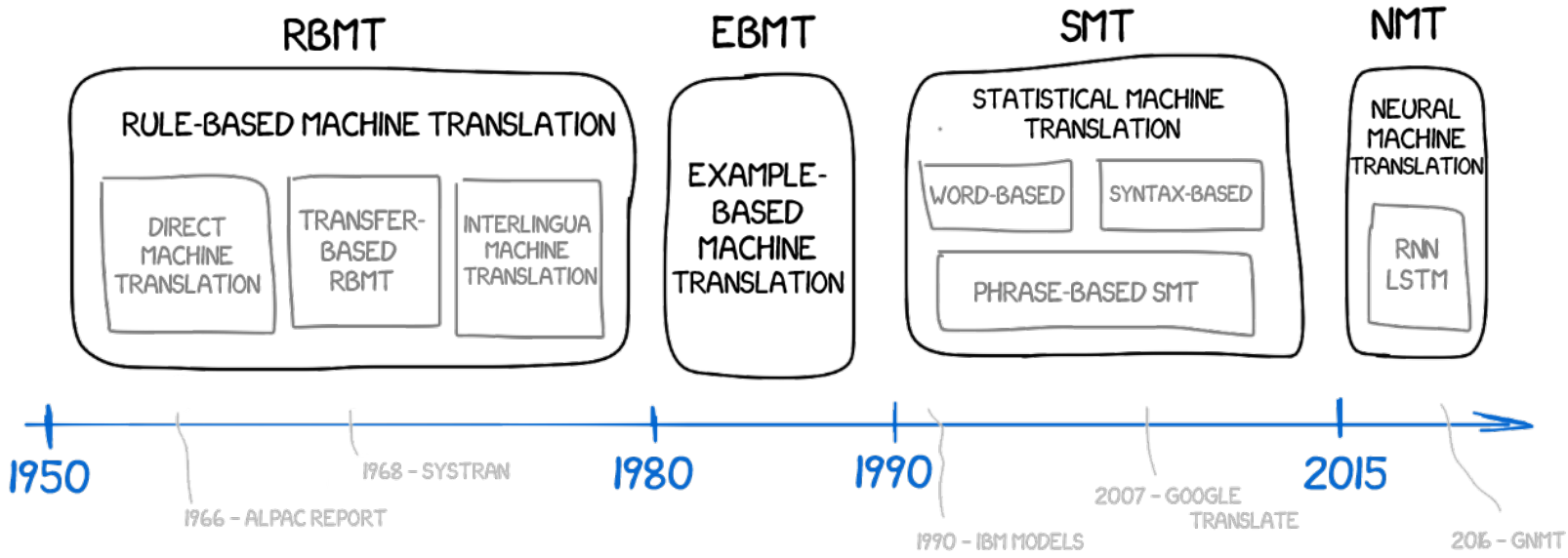
Machine Translation

Generative process: breaking up translation process into smaller steps

→ sentences are sequences of lexical units (i.e. words)

→ translating a word = assigning a word in target language

Machine Translation – Timeline



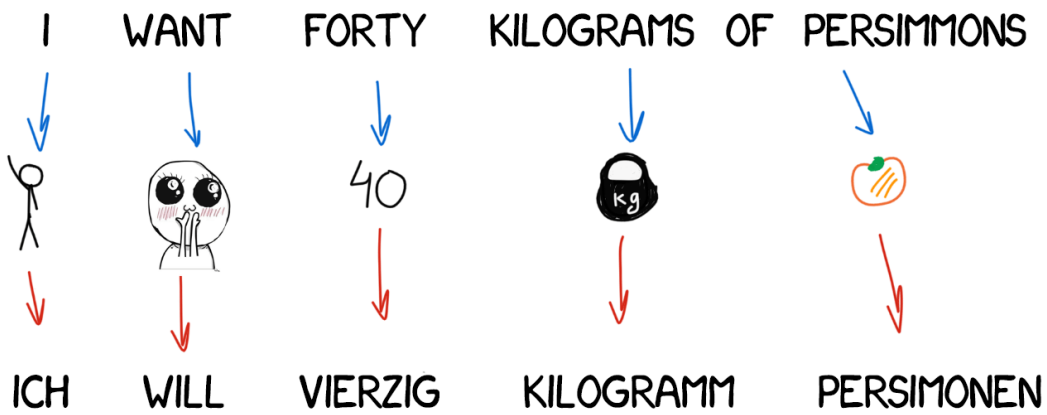
Rule-Based Machine Translation

Rule-based Machine Translation (RBMT)

- **DIRECT** Machine Translation
  - Bilingual dictionary (e.g. Russian into English)
  - Set of linguistic rules for each language

- **TRANSFER**-based Machine Translation
  - Morphological and syntactic analysis (i.e. more complex than Direct MT)

- **INTERLINGUAL** Machine Translation



Rule-Based Machine Translation (RBMT)

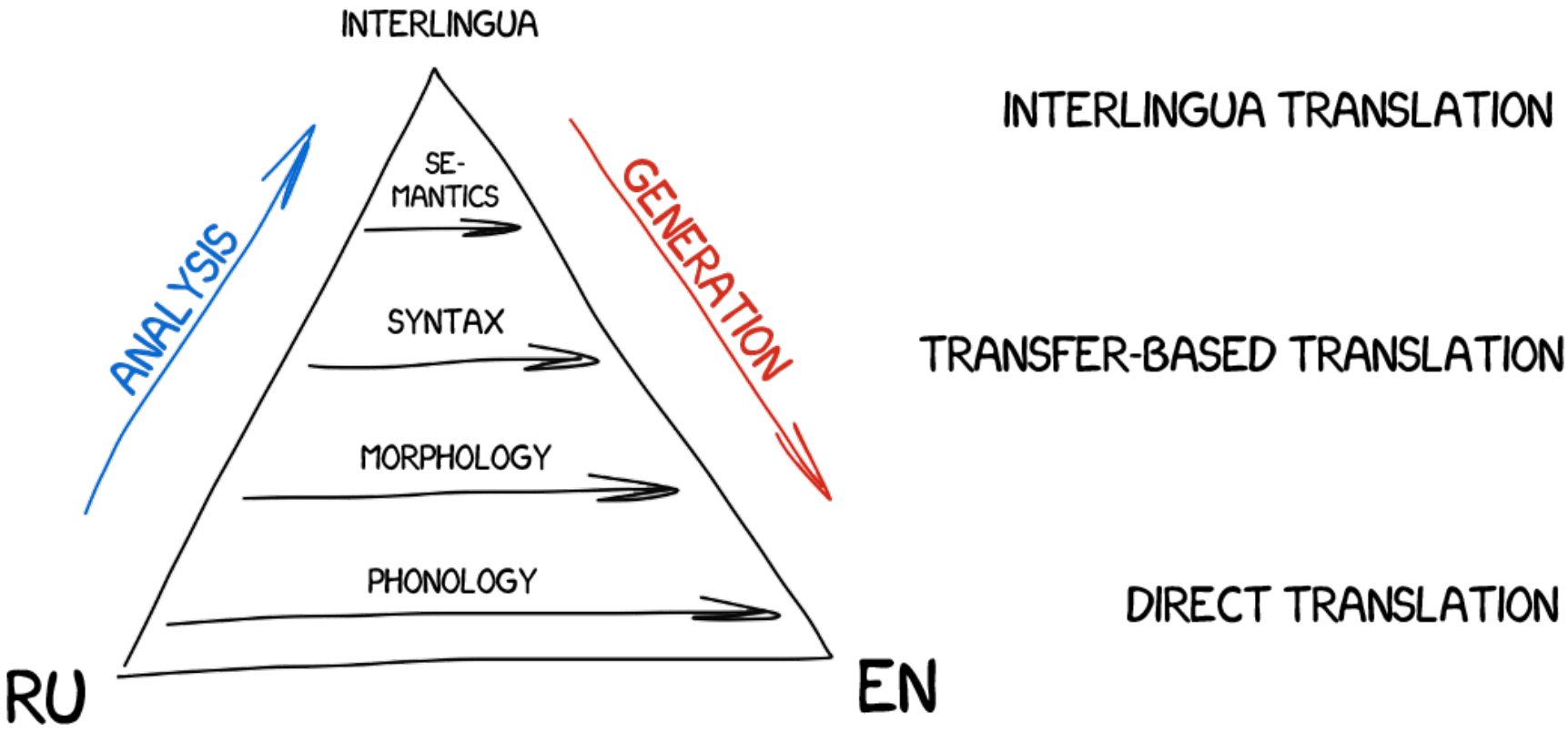


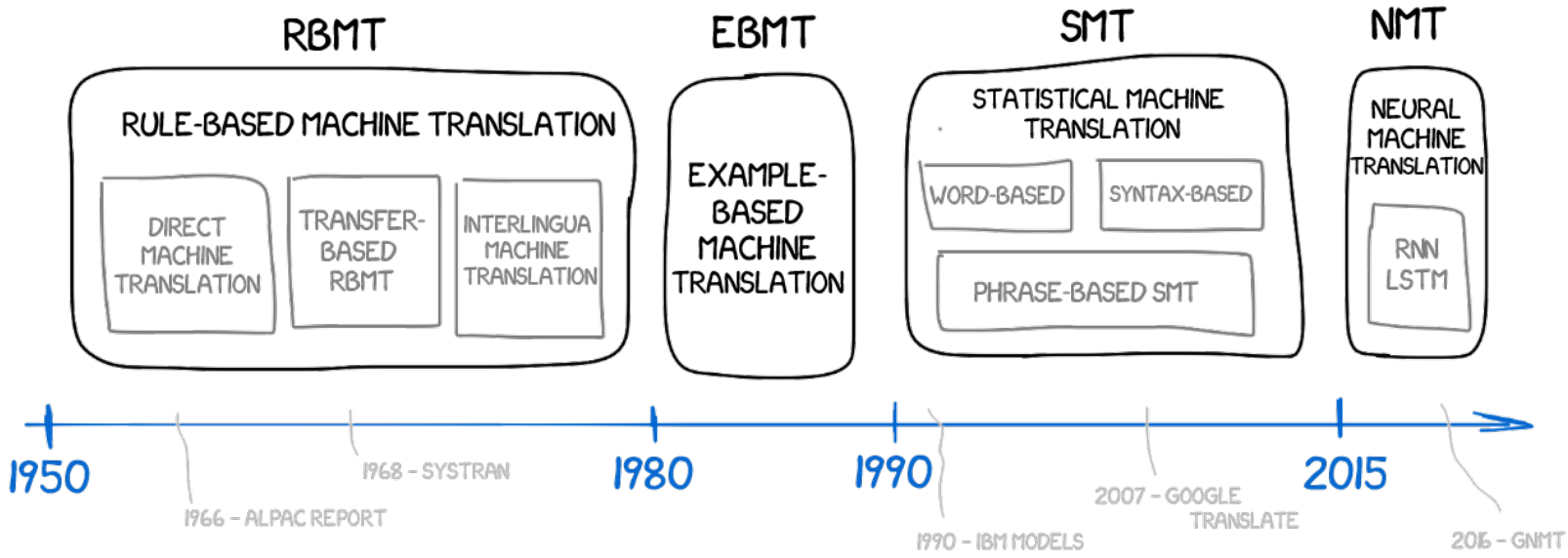
Fig. The Vauquois Triangle

"The Rule-based approach will attempt to provide a full modeling of the car dynamic, on how the engine is connected to the wheel, on the effect of acceleration in the trajectory, etc. This is very complicated (and possibly impossible to model in totality)."

— **J. Senellart (2016)**

Global CTO at SYSTRAN

Timeline of Machine Translation

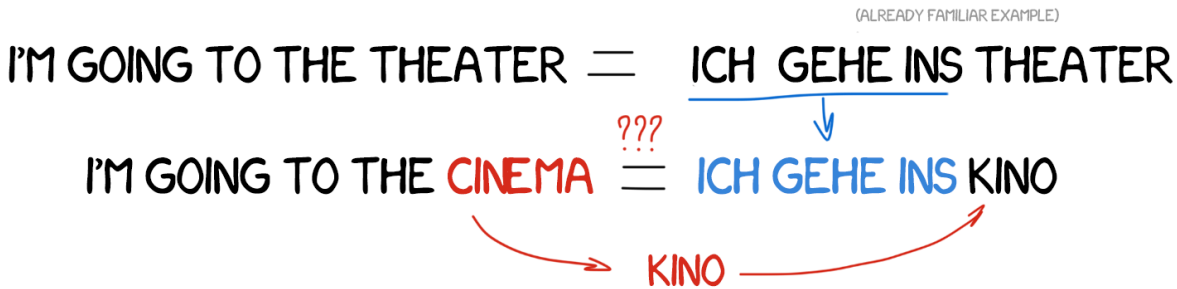


Data-driven methods (1980-201?)

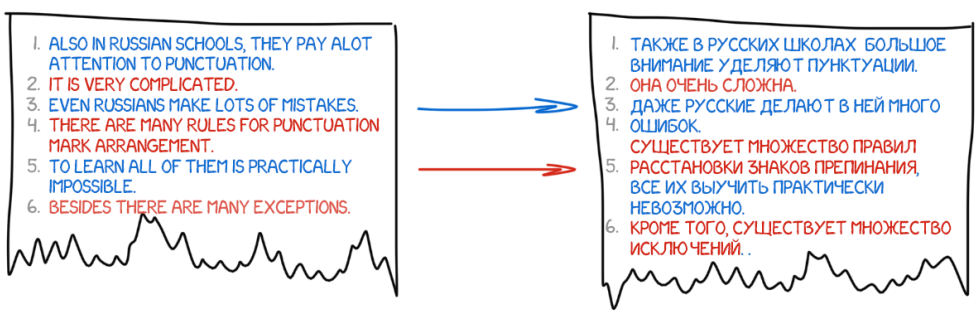
Data-driven methods (1980-201?)

→ Learn how to translate from past translation examples!

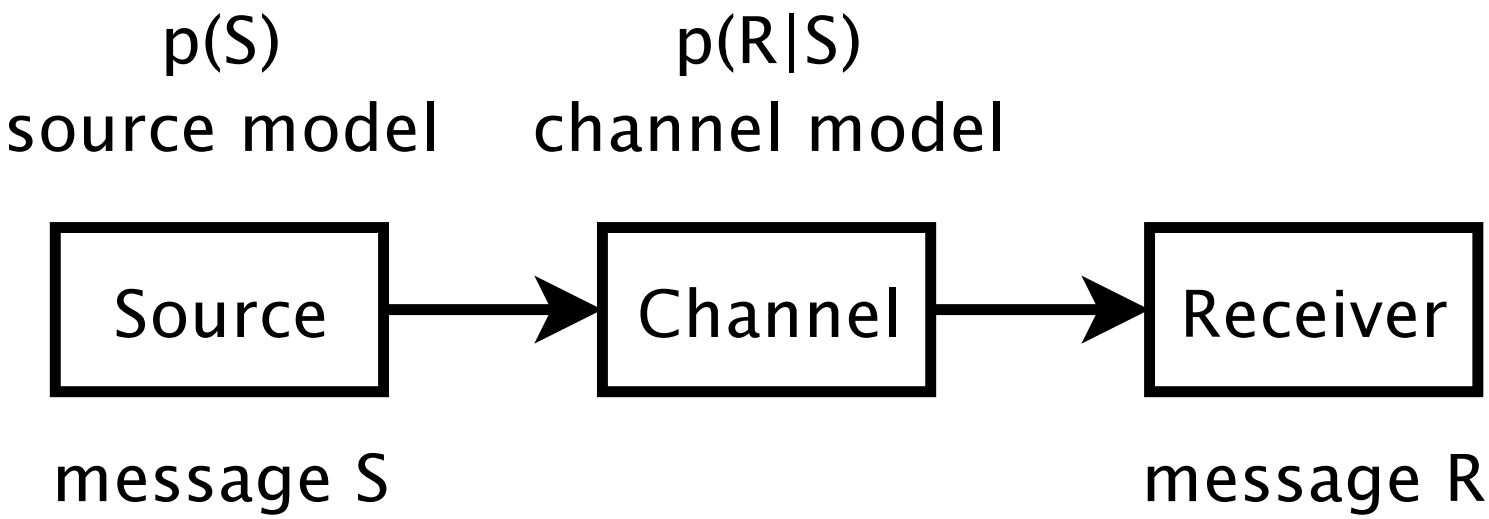
- **EXAMPLE**-based Machine Translation (EBMT)



- **STATISTICAL**-based Machine Translation (SMT)



Data-driven methods (1980-201?) – Focus on SMT



- we observe a distorted message R (e.g. foreign string **f**)
- **we have a model on how the message is distorted (*a.k.a.* translation model)**
- **we have a model on what messages are probably (*a.k.a.* language model)**
- we want to recover the original message S (e.g. English string **e**)

Data-driven methods (1980-201?) – Focus on SMT

Derivation of "noisy channel model" in a probabilistic framework using the Bayes rule:

$$\begin{aligned} \text{argmax}_{\mathbf{e}}\ p(\mathbf{e}|\mathbf{f}) &= \text{argmax}_{\mathbf{e}} \frac{p(\mathbf{f}|\mathbf{e})\ p(\mathbf{e})}{p(\mathbf{f})} \\ &= \text{argmax}_{\mathbf{e}}\ p(\mathbf{f}|\mathbf{e})\ p(\mathbf{e}) \end{aligned}$$

- $p(\mathbf{e})$: language model – fluency in the target language
- $p(\mathbf{f}|\mathbf{e})$: translation model – **lexical translation probabilities**

Data-driven methods (1980-201?) – Focus on SMT

Derivation of "noisy channel model" in a probabilistic framework using the Bayes rule:

$$\begin{aligned} \text{argmax}_{\mathbf{e}}\ p(\mathbf{e}|\mathbf{f}) &= \text{argmax}_{\mathbf{e}} \frac{p(\mathbf{f}|\mathbf{e})\ p(\mathbf{e})}{p(\mathbf{f})} \\ &= \text{argmax}_{\mathbf{e}}\ p(\mathbf{f}|\mathbf{e})\ p(\mathbf{e}) \end{aligned}$$

- $p(\mathbf{e})$: language model – fluency in the target language
- $p(\mathbf{f}|\mathbf{e})$: translation model – **lexical translation probabilities**

→ Cf. Lecture #3: "Language Modeling"
→ Cf. Lecture #2: "Sequence Labelling and Part-of-Speech Tagging"

Lexical Translation Probabilities

Lexical Translation Probabilities

- How to translate a word? $\rightarrow$ look up into a dictionary
  ex: **haus** — *house, building, home, household, shell.*
- Multiple translations (some more frequent than others)
  - for instance: house, and building most common
  - special cases: haus of a snail is its shell

Translation of *haus*	**Count**
house	8,000
building	1,600
home	200
household	150
shell	50

Lexical Translation Probabilities

From the word frequencies, we can estimate a **lexical translation probability distribution:**

$$p_f(e) = \begin{cases} 0.8 & \text{if } e = \text{house}, \\ 0.16 & \text{if } e = \text{building}, \\ 0.02 & \text{if } e = \text{home}, \\ 0.015 & \text{if } e = \text{household}, \\ 0.005 & \text{if } e = \text{shell}. \end{cases}$$

also called **maximum likelihood estimation**

– Question –

How do we obtain those counts?

Tip – counts are observations made over corpus aligned at sentence-level...

– Question –

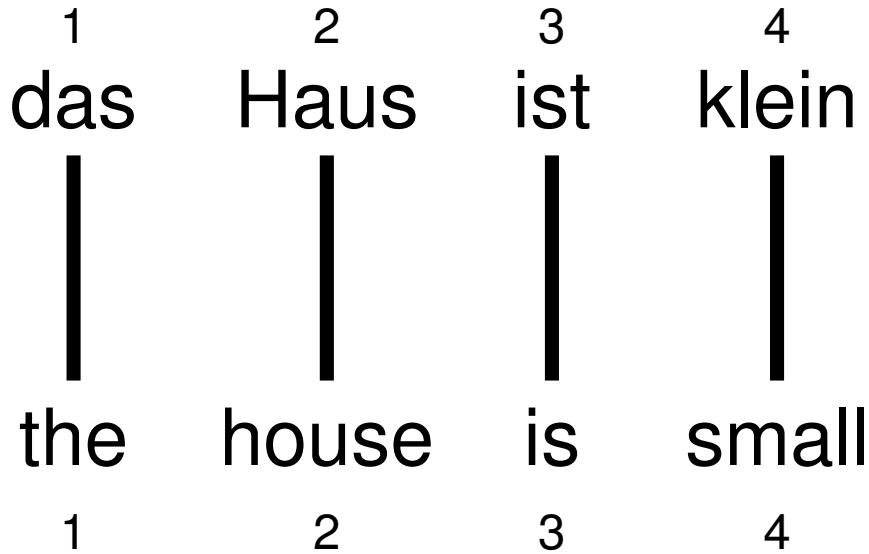
How do we obtain those counts?

Tip – counts are observations made over corpus aligned at sentence-level...

Alignments between words

Lexical Translation Probabilities

- when we **translate**, we **align** the words in one language, with the words in the other:



here we have 1 to 1 alignments, and word positions are numbered 1–4

Lexical Translation Probabilities

What do we want?

- formalizing alignments at word-level, with an **alignment function** a, mapping a target word at position i, to a source word at position j, such as: $a : i \rightarrow j$

Alignment function from previous example:

$$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4\}$$

IBM Model 1

Model that generates a number of different translations for a sentence, each with a different probability:

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

- for a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$ of length l_f
- to an English sentence $\mathbf{e} = (e_1, ..., e_{l_e})$ of length l_e
- with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \rightarrow i$
- parameter ϵ is a normalization constant

Example

das

e	$t(e\|f)$
the	0.7
that	0.15
which	0.075
who	0.05
this	0.025

Haus

e	$t(e\|f)$
house	0.8
building	0.16
home	0.02
household	0.015
shell	0.005

ist

e	$t(e\|f)$
is	0.8
's	0.16
exists	0.02
has	0.015
are	0.005

klein

e	$t(e\|f)$
small	0.4
little	0.4
short	0.1
minor	0.06
petty	0.04

$$p(e, a|f) = \frac{\epsilon}{5^4} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein})$$
$$= \frac{\epsilon}{5^4} \times 0.7 \times 0.8 \times 0.8 \times 0.4$$
$$= 0.001344\epsilon$$

– Question –

How do we learn these lexical translation probabilities?

So far we assumed that we already have them...

Chicken and egg problem:

- **if we had the *alignments*, we could estimate the *parameters* of our generative model**
- **if we had the *parameters*, we could estimate the *alignments***

– Question –

How do we learn these lexical translation probabilities?

So far we assumed that we already have them...

Chicken and egg problem:

- **if we had the *alignments*, we could estimate the *parameters* of our generative model**
- **if we had the *parameters*, we could estimate the *alignments***

Expectation Maximization Algorithm

Expectation Maximization (EM) Algorithm

This algorithm is an iterative learning method, that addresses the situation of **incomplete data**:

- if we had *complete data*, we could estimate our *model*
- if we had our *model*, we could fill in the *gaps in the data* (here, to find the most likely alignments between words)

- EM in a nutshell
  1. initialize the model, typically with uniform distributions;
  2. assign probabilities to the missing data;
  3. estimate model parameters from completed data;
  4. iterate steps 2–3 until convergence.

– Question –

What happen if our translation model cannot decide between two words (e.g. small and little)?

- **Sometime one is preferred over the other**
  - **small step: 3,150,000,000 occurrences in the Google index**
  - **little step: 2,650,000,000 occurrences in the Google index**

– Question –

What happen if our translation model cannot decide between two words (e.g. small and little)?

- **Sometime one is preferred over the other**
  - **small step: 3,150,000,000 occurrences in the Google index**
  - **little step: 2,650,000,000 occurrences in the Google index**

Language Model

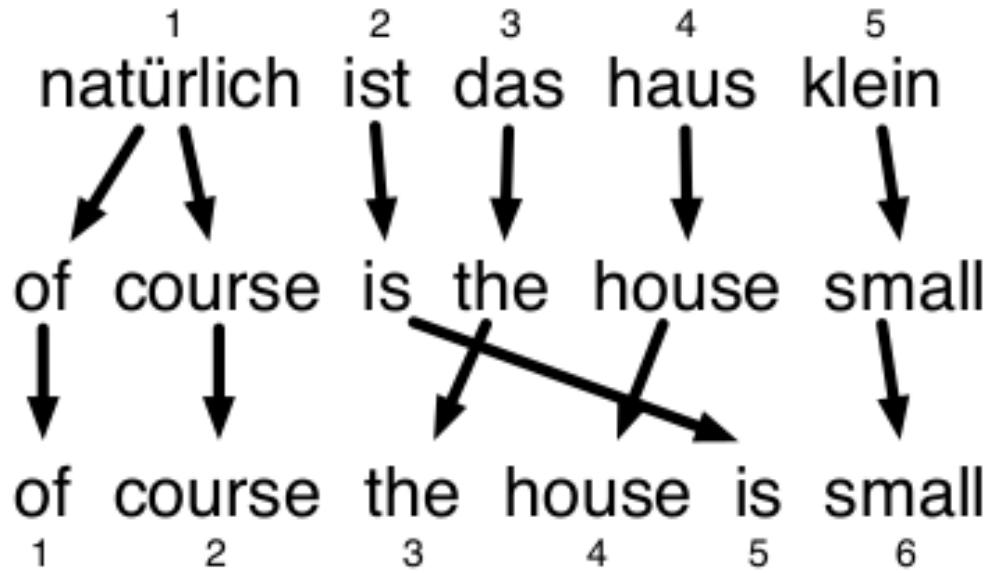
– Question –

Is IBM Model 1 a good model? and why?

IBM Model 2

Considering the word order in sentences: we add an explicit model for alignment

- 1st step: lexical translation (i.e. IBM Model 1)
- 2nd step: alignment



IBM Model 2

Modeling alignment with an **alignment probability distribution**:

- Translating word f at position i, to word e at position j:

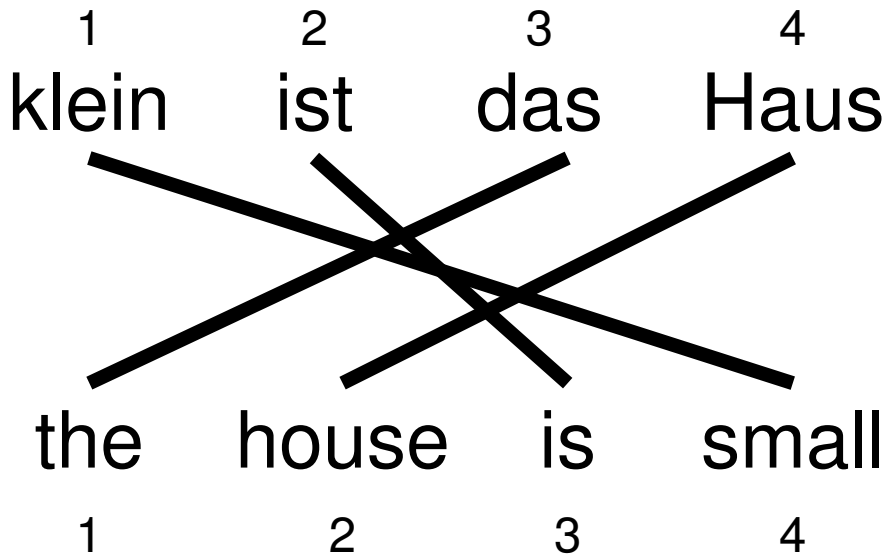
$$a(i|j, l_e, l_f)$$

- Putting everything together

$$p(\mathbf{e}, a|\mathbf{f}) = \epsilon \prod_{j=1}^{l_e} t(e_j|f_{a(j)})\ a(a(j)|j, l_e, l_f)$$

Reordering

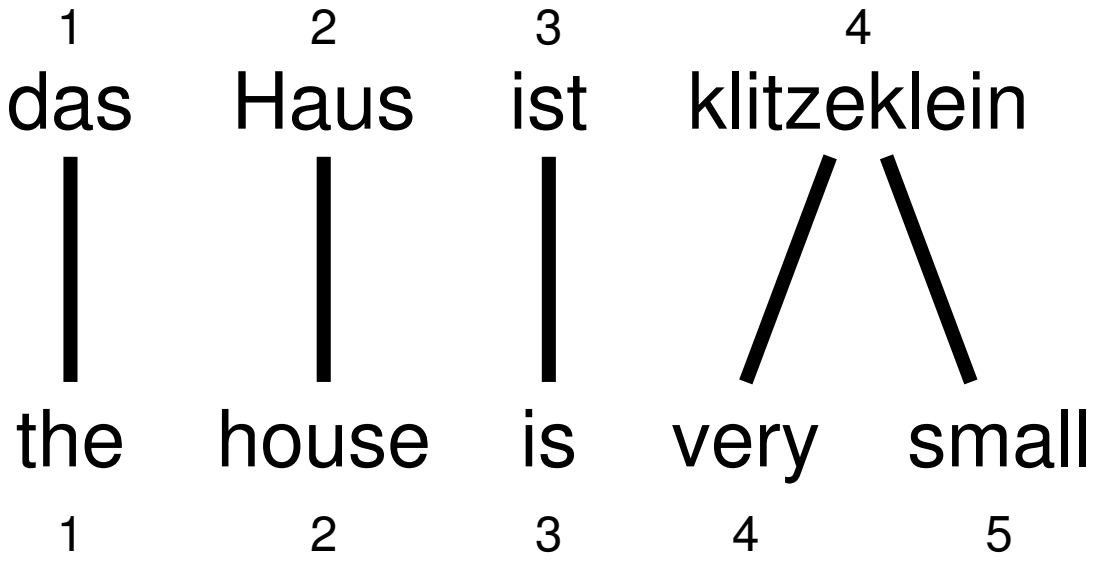
IBM Model 1 considers all possible reordering as equally likely. Often, words that follows each other in one language have translations that follow each other in the output language



$a : \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\}$

One-to-Many Translation

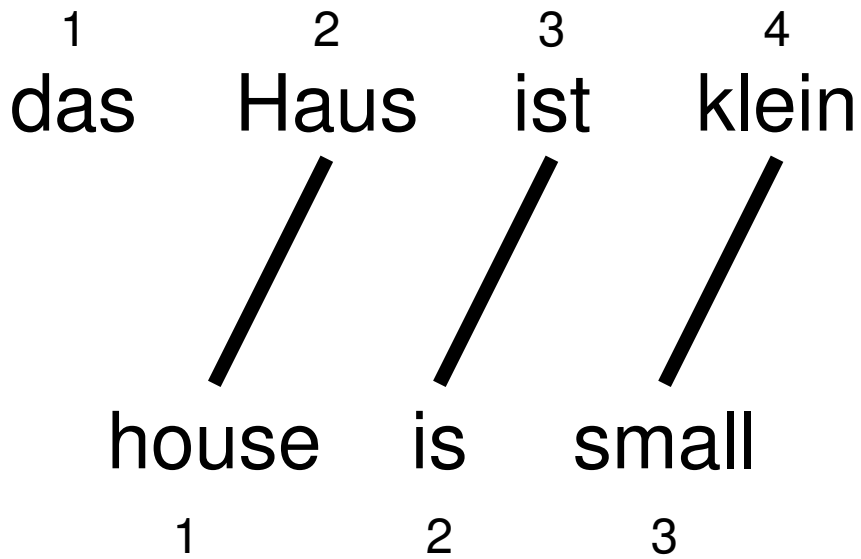
A source word may translate into multiple target words (a.k.a **fertility**)



$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4, 5 \rightarrow 4\}$

Dropping Words

Words may be dropped when translated (e.g. the German article das is dropped)

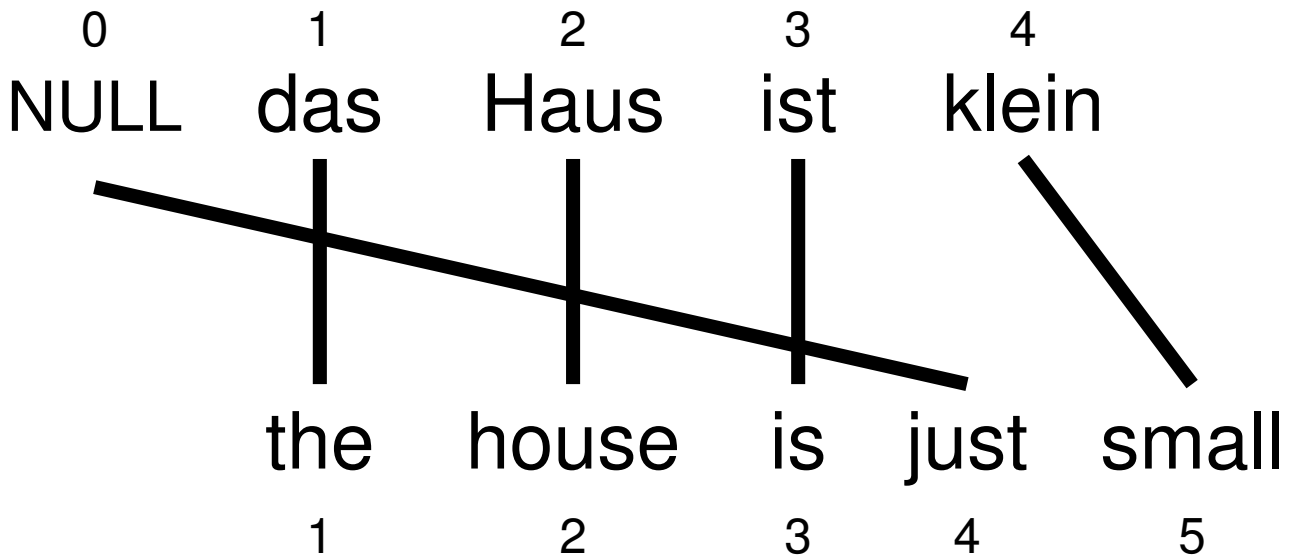


$a : \{1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4\}$

Inserting Words

Words may be added during translation

- the English just does not have an equivalent in German
- we still need to map it to something: special **null** token



$$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 0, 5 \rightarrow 4\}$$

Higher IBM Models

IBM Model 1	lexical translation
IBM Model 2	adds absolute reordering model
IBM Model 3	adds fertility model
IBM Model 4	relative distorsion model
IBM Model 5	fixes deficiency

IBM 4: the placement of the translation of an input word is typically based on the placement of the translation of the preceding input word (improves IBM 3!)

IBM 5: for IBM 3 & 4, mulitple output words may be placed in the same position (not possible!)

Data-driven methods (1980-201?) – Summary

Focus on word-based Model

- Noisy Channel Model
- Lexical translation probabilities
- Alignments at word-level
- Expectation Maximization (EM) Algorithm
- IBM Models 1–5
  - IBM Model 1: lexical translation
  - IBM Model 2: alignment model
  - —
  - IBM Model 3: fertility
  - IBM Model 4: relative alignment model
  - IBM Model 5: deficiency

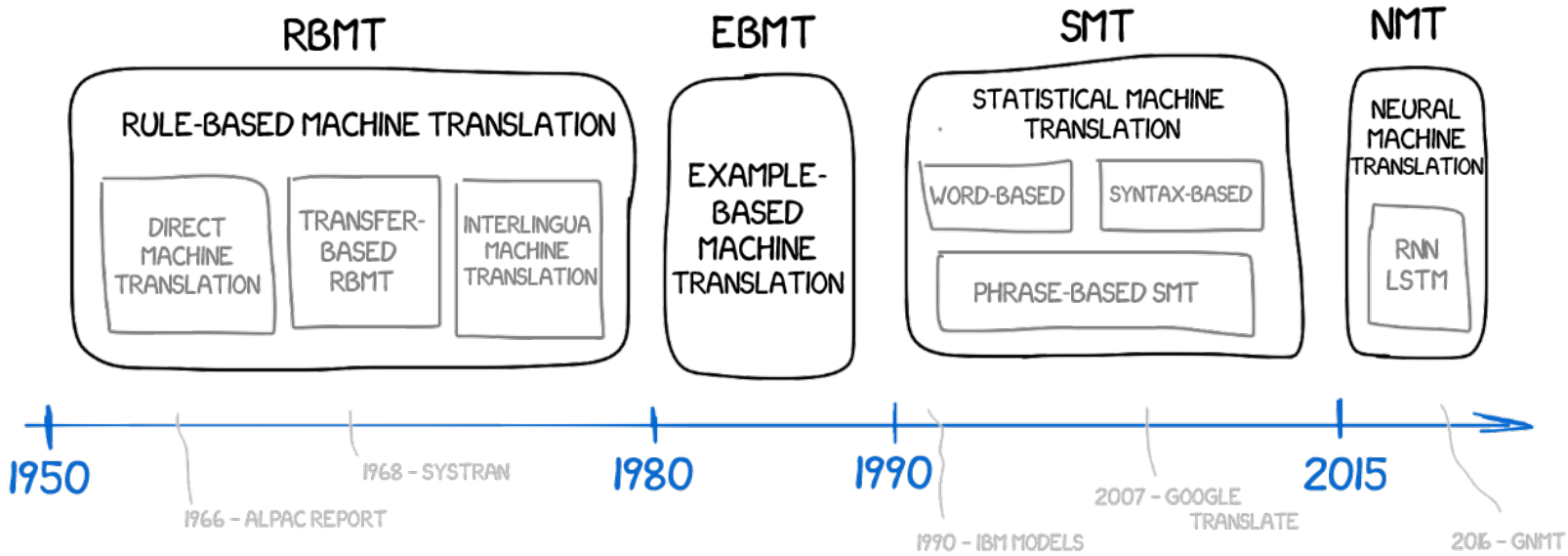
Does not constitute the state-of-the-art anymore, but many of the principles and methods are still valid today!

"The Statistical approach, will use data from past experience and will try to compare a new situation with a past situation and will decide on the action based on this large database. This is a huge task and very difficult to implement (and can only be as good as the database it learns from)."

— J. Senellart (2016)

Global CTO at SYSTRAN

Timeline of Machine Translation



Neural Machine Translation

"The Neural approach, with a limited access to the phenomenon involved, or with limited ability to remember, will build its own "thinking system to optimize the driving experience, it will actually learn to drive the car, build reflexes – but will not be able to explain why and how such decisions are being made [...]"

— J. Senellart (2016)

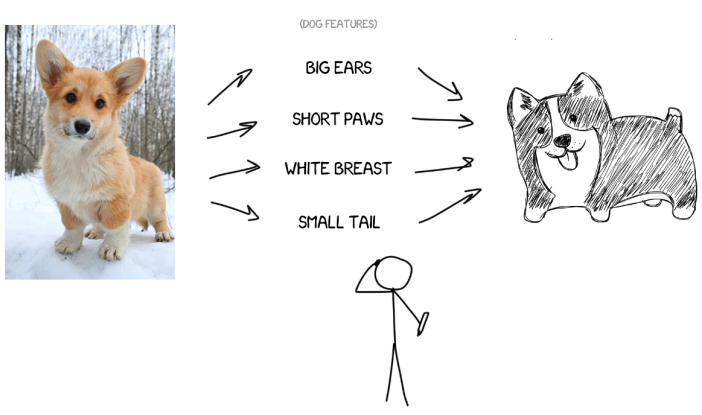
Global CTO at SYSTRAN

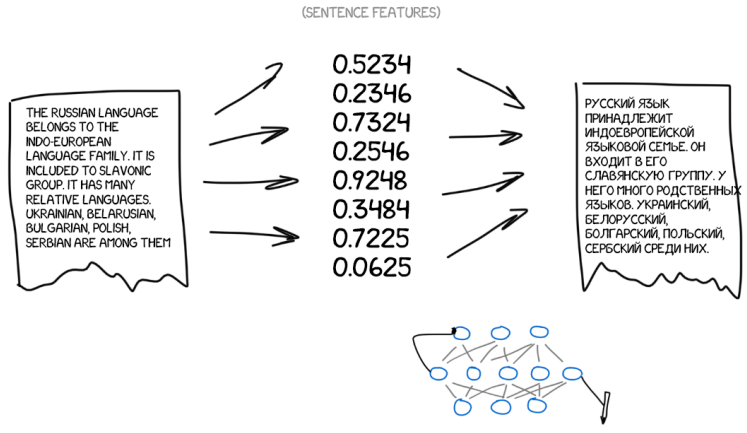
Neural Networks

Example: Prisma, mobile app that allows you to transfer the style of one image, onto the content of another

→ if we can transfer a style to a photo, can we impose another language to a source text?

Neural Machine Translation (NMT)





Neural-based approaches $\Rightarrow$ **Current State-Of-The-Art in MT!**

– Question –

How to encode a sequence of words into a vector?

Tip – Nikos' lecture last week ;)

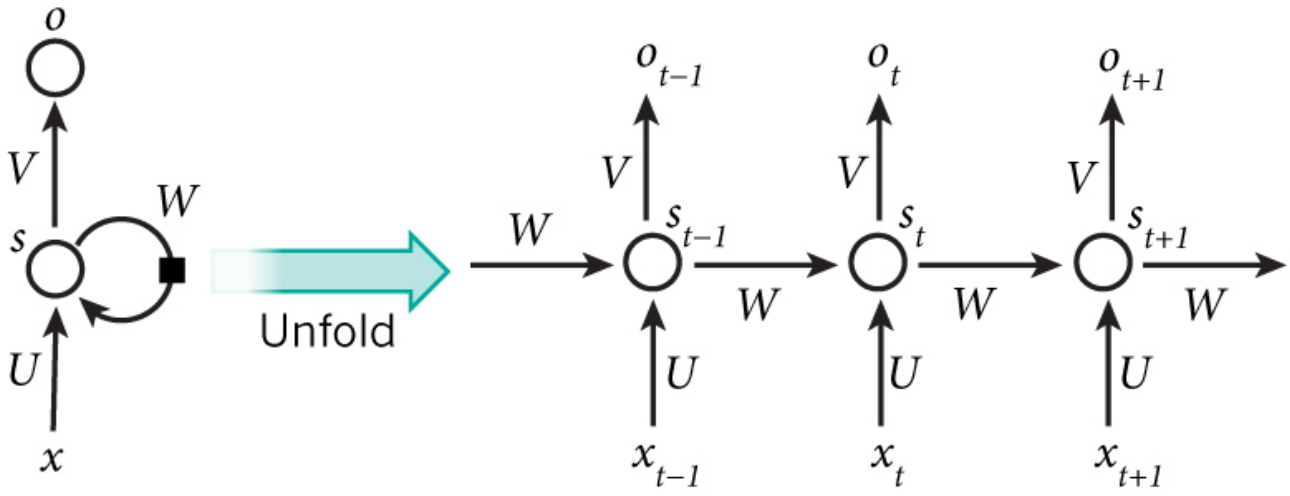
– Question –

How to encode a sequence of words into a vector?

Tip – Nikos' lecture last week ;)

Recurrent Neural Networks

Recurrent Neural Networks (recap)

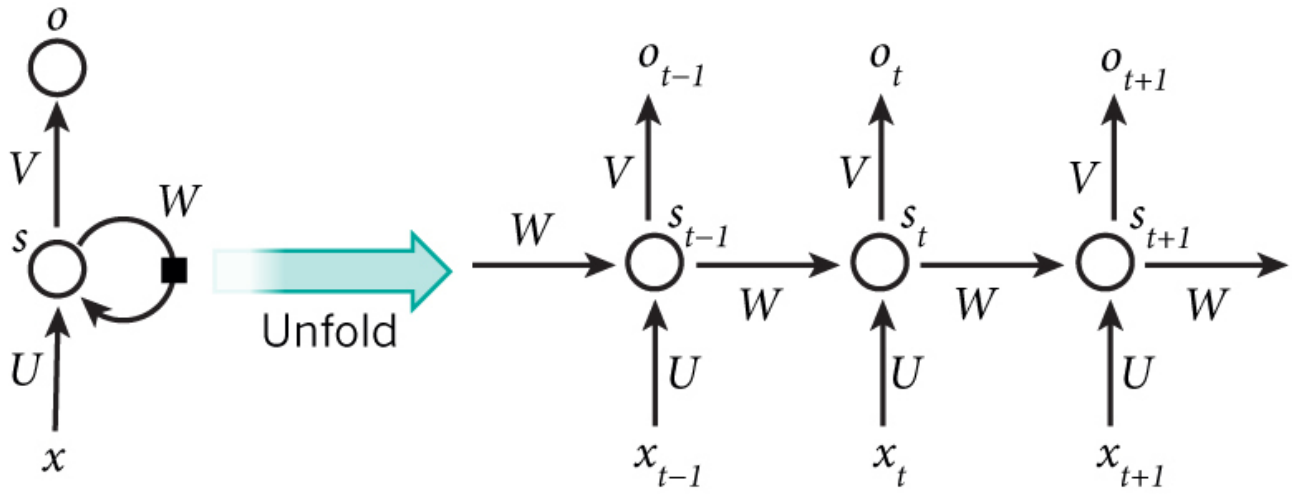


At each time step:

- takes two input vectors, produces two output vectors
- calculates a new hidden state ("memory")

Why recurrent?

Recurrent Neural Networks (recap)



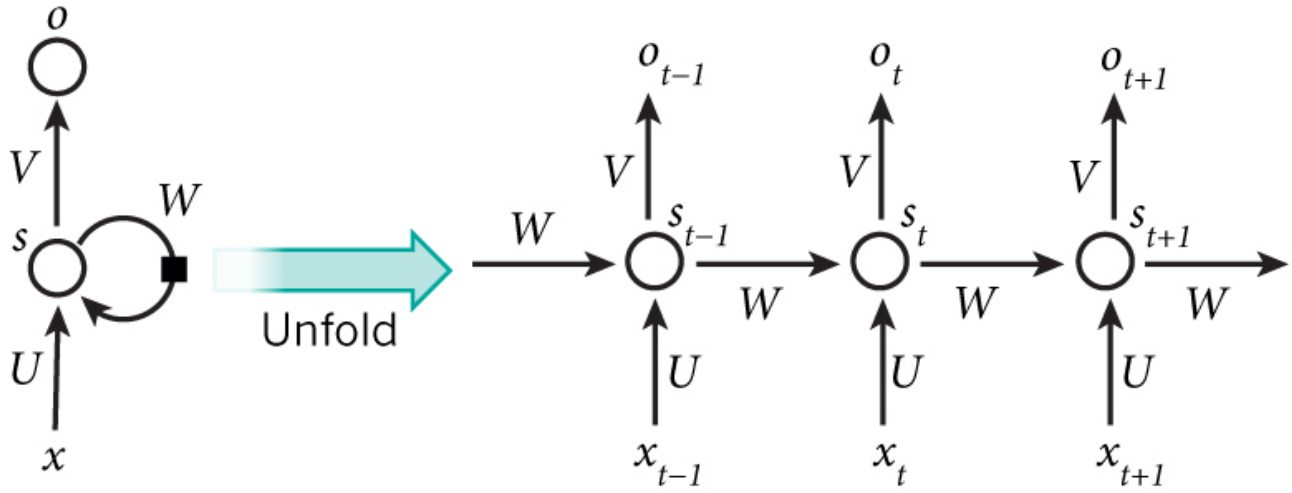
At each time step:

- takes two input vectors, produces two output vectors
- calculates a new hidden state ("memory")

Why recurrent?

- **same parameters** (U, W, V) shared at each time step, but calculation based on different inputs

Recurrent Neural Networks for Neural MT



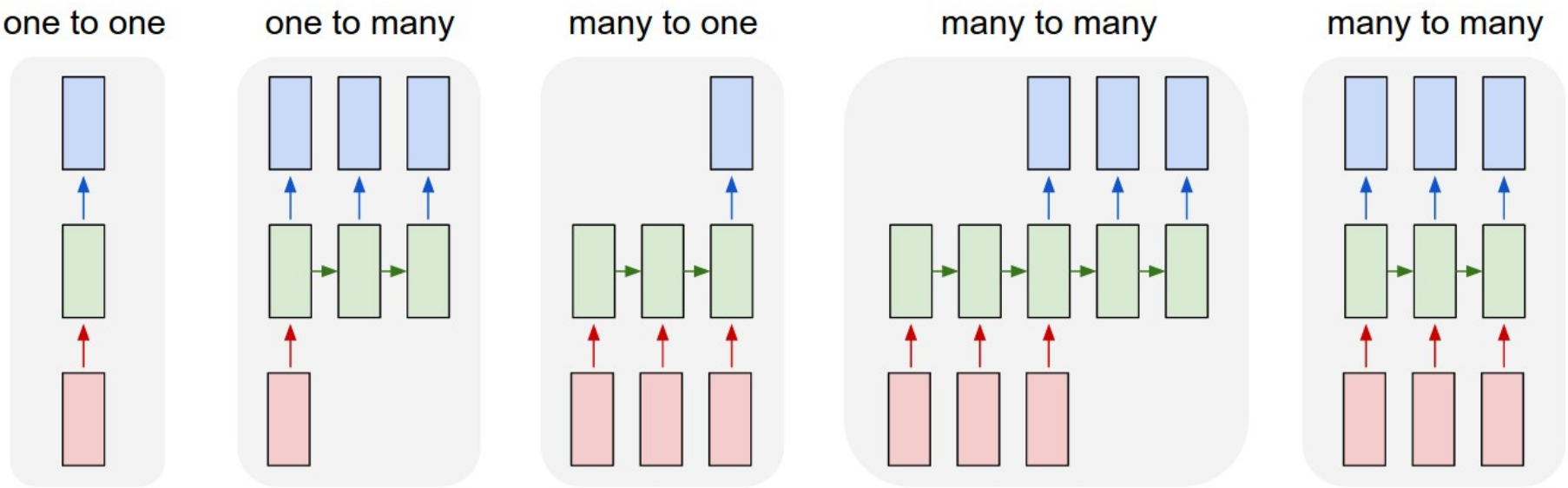
Vector representation of an input sequence x:

$$\mathbf{c = RNN(x)}$$

Probability of the target sequence y|x:

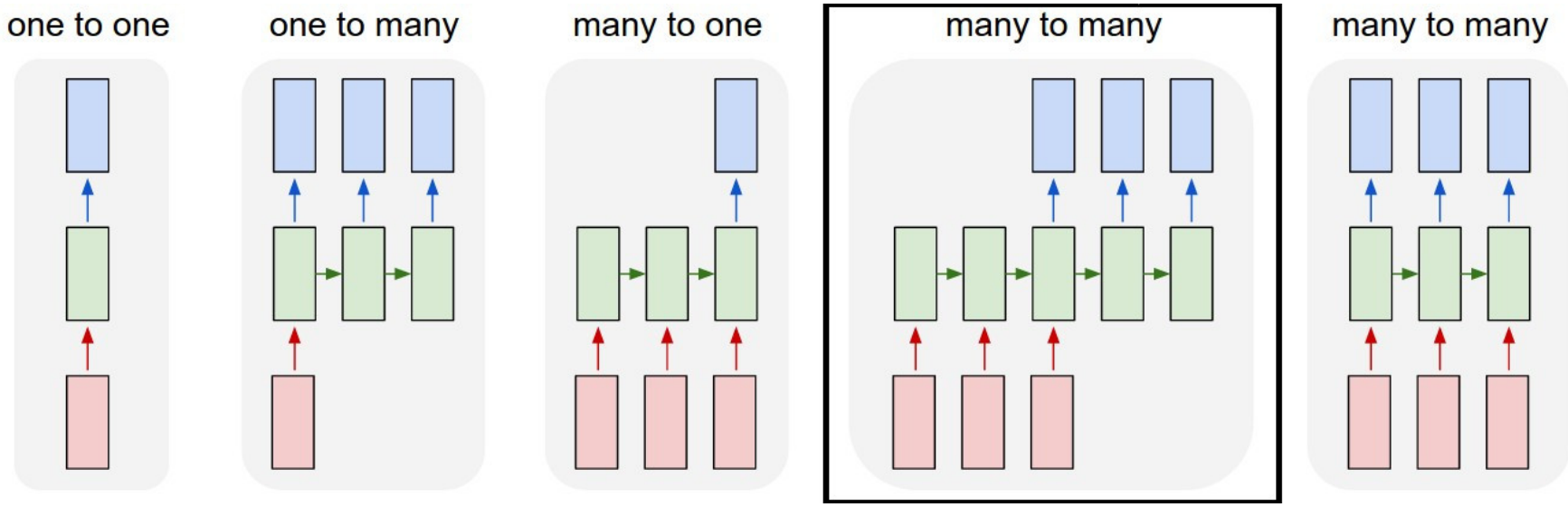
$$\mathbf{y|x \sim RNNLM(c)}$$

Recurrent Neural Networks for Neural MT



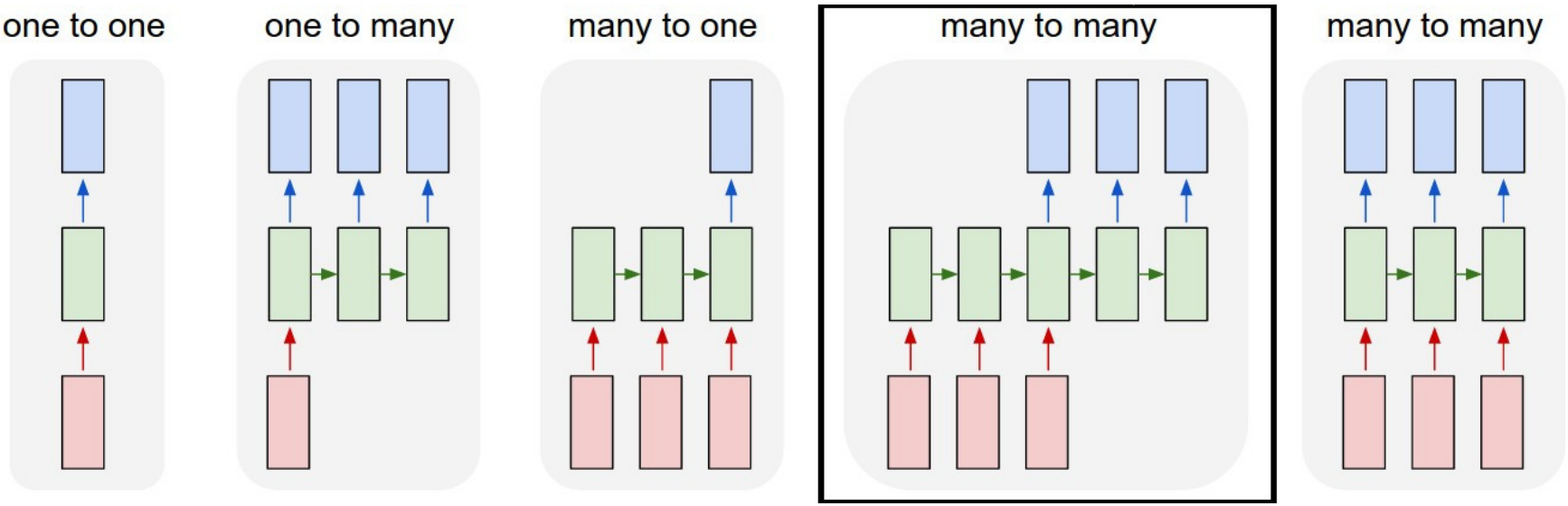
- many to one: *e.g.* text classification
- many to many (equal): *e.g.* PoS tagging

Recurrent Neural Networks for Neural MT



- many to one: *e.g.* text classification
- many to many (equal): *e.g.* PoS tagging
- many to many (unequal): Machine Translation

Recurrent Neural Networks for Neural MT



- many to one: *e.g.* text classification
- many to many (equal): *e.g.* PoS tagging
- many to many (unequal): Machine Translation

$\hookrightarrow$ **Neural sequence-to-sequence models for MT**

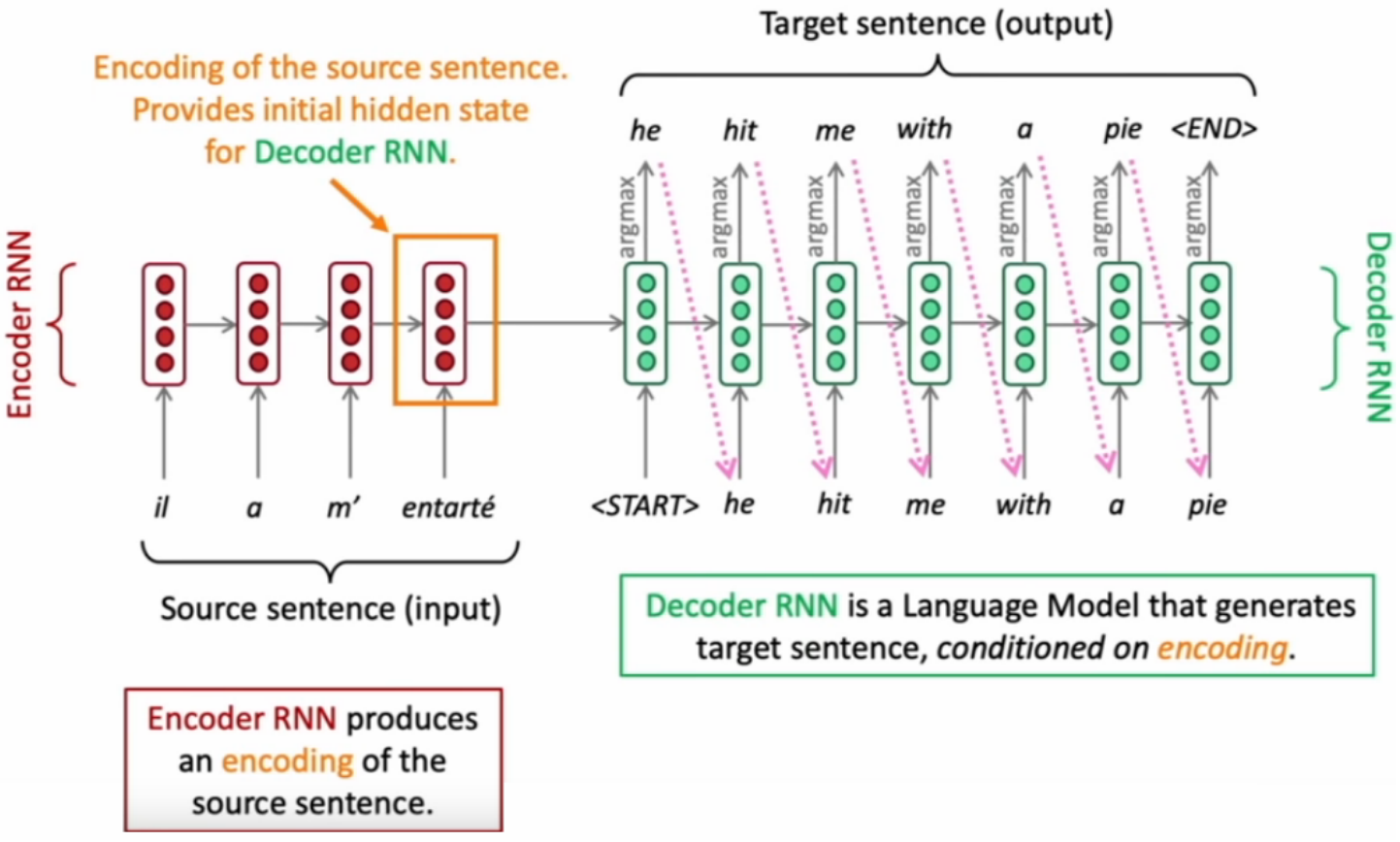
Neural Machine Translation

→ Conditional Language Model

- Language Model because the decoder is predicting the next word of the target sentence
- Conditional because its predictions are also conditioned on the source sentence
- Neural MT directly calcules $p(\mathbf{y}|\mathbf{x})$:

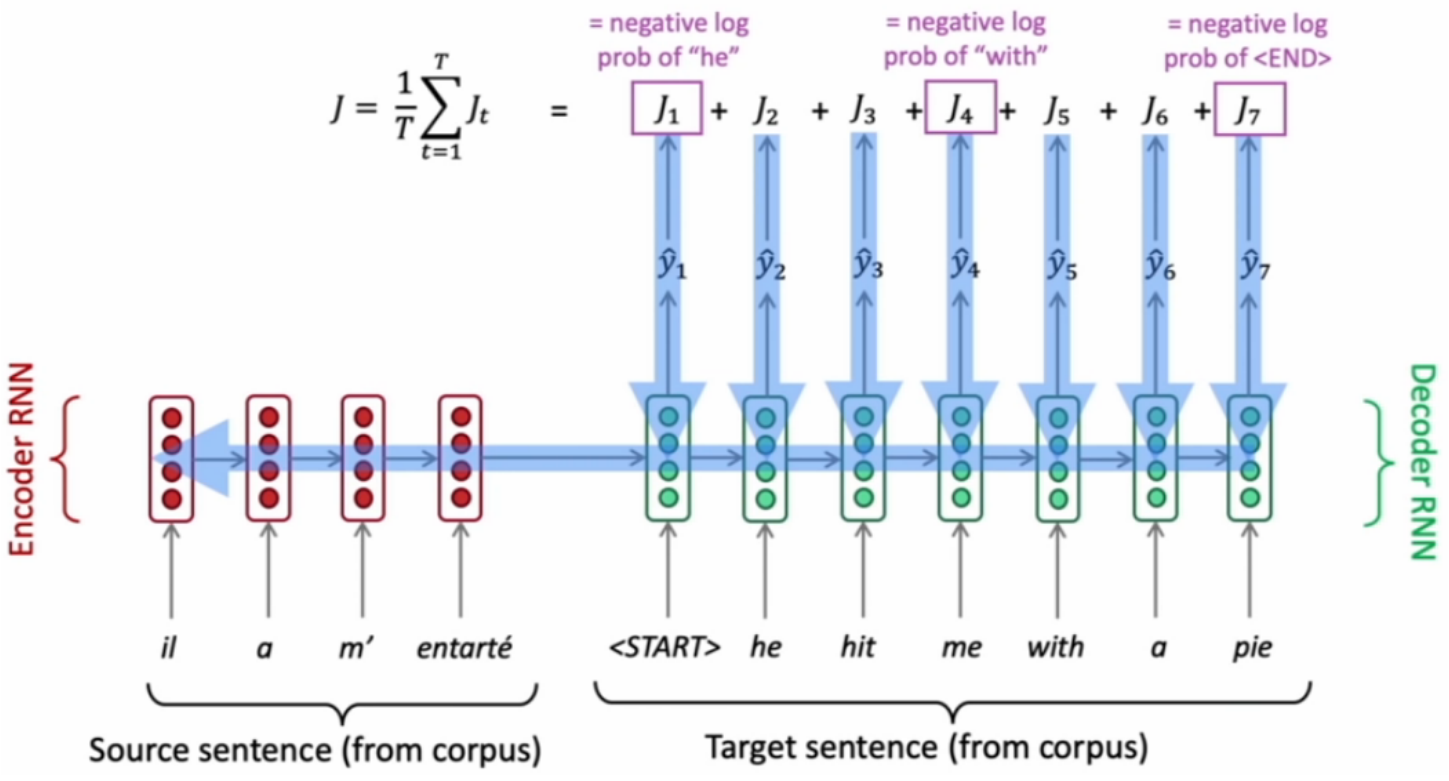
$$p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y_1}|\mathbf{x})p(\mathbf{y_2}|\mathbf{y_1}, \mathbf{x})p(\mathbf{y_3}|\mathbf{y_3}, \mathbf{y_2}, \mathbf{x}) \dots p(\mathbf{y_T}|\mathbf{y_1}, \dots, \mathbf{y_{T-1}}, \mathbf{x})$$

Neural Machine Translation



→ Neural Seq2Seq MT is trained "end-to-end"

- network optimized by considering the inputs and outputs directly



Neural Machine Translation

Which strategy during decoding:

- **greedy** search? which leads to **error propagation**
  - ↪ garden-path problem: sometimes we follow a sequence of words and realize too late that we made a mistake early on...

- **exhaustive** search? compute all possible sequences (**too expensive!**)
  - ↪ At each step t of the decoder, we are tracking V^t possible partial translations, where V is vocab size
  - ↪ This $O(V^T)$ is far **too expensive**

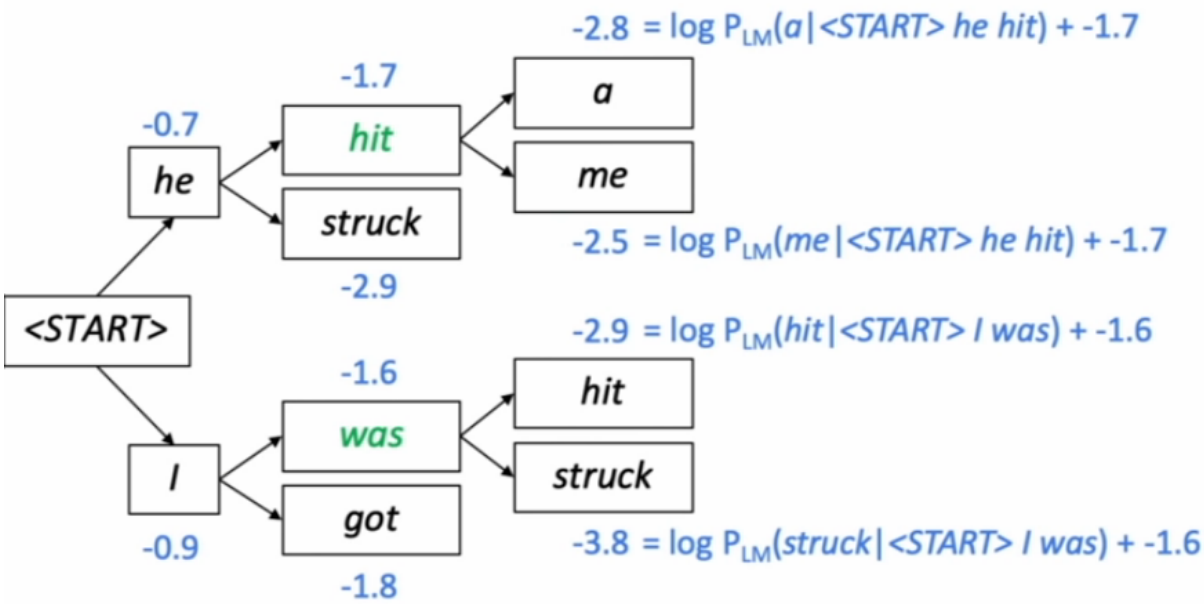
Neural Machine Translation

Which strategy during decoding:

use an approximate search algorithm: **Beam search decoding**

↪ for each of the k hypotheses, find top k next words and calculate score (common practive is k between 5 and 10)

↪ does not guarantee to find to find optimal solution but more efficient the exhaustive search



Advantages of Neural MT (compared to SMT)

Better performance

- Improved fluency (which can be lead to problems...)
- Better use of context
- Better use of phrase similarity

A single neural network to be optimized end-to-end

- No subcomponents to be individually optimized

Less feature engineering effort

- Same method for all language pairs

– Question –

Is Machine Translation solved?

– Question –

Is Machine Translation solved?

Nope!

Disadvantages of Neural MT (compared to SMT)

Less interpretable

- Errors are (very) hard to track/debug

Difficult to control

- Hard to constrain the system to impose what you want!

Requires huge computation power to be trained

- GPUs required!

Is Machine Translation solved?

Out-of-vocabulary words

- input side: embedding matrix that maps each word into its embedding
- output side: predict a probability distribution over all output words

$\rightarrow$ amount of computation **linear** with the size of the vocabulary!

$\rightarrow$ Opt #1: restrict the vocabulary to the most common words, replace the other with 'UNK', and translate rare words using a back-off dictionary

$\rightarrow$ Opt #2: break up rare words into **subword units**, similar to standard approach in Statistical MT to handle compound *e.g.* website $\rightarrow$ web + site)

Is Machine Translation solved?

Domain mismatch between train and test data

- train a huge amount of data, **hard to shift** from one domain to another

Some languages are well-resourced, some others under resourced

- use **synthetic data** by back translation
  $\hookrightarrow$ translate target side monolingual data with a reverse MT model, combine true and synthetic parallel data to train the final model

Is Machine Translation solved?

Is Neural MT output trustworthy?

- Recent studies have shown that users rely more on **fluency** than **adequacy** to trust MT
- Translation that **looks fine at first sight**, may in reality contain inadequate words that won't be detected by the user

  $\rightarrow$ active field of research, check **Quality Estimation**!

$\rightarrow$ **Research on Neural MT is a very active!**

– Question –

What can possibly be the limitations of RNNs?

Tip – the NMT decoder is conditioned by the last hidden state of the encoder

– Question –

What can possibly be the limitations of RNNs?

Tip – the NMT decoder is conditioned by the last hidden state of the encoder

Can't capture long-range dependencies, information bottleneck

Recurrent Neural Networks for Neural MT

We ask a lot to the hidden state

- During encoding:
  - needs to combine information about the **all sentence**
  - should not **forget** the first words when encoding towards the end of the sentence

Minor refinement: reverse the order of the output sentence: the last encoded words are close to the last words of the translation

We ask a lot to the hidden state

- During decoding:
  - has to help **predicting** each word
  - needs to **know** what have already been translated, and what still needs to be translated

Works reasonably well for sentences up to 10–15 words

Minor refinement: use the sentence embedding state as input to all hidden states of the decoder phase (no need to remember the input anymore!)

– Question –

Still not enough... why??

Tip – ask Pr. Ray Mooney from The University of Texas!

– Question –

Still not enough... why??

Tip – ask Pr. Ray Mooney from The University of Texas!

Sentences have unbounded length, vectors have fixed sizes

Recurrent Neural Networks for Neural MT

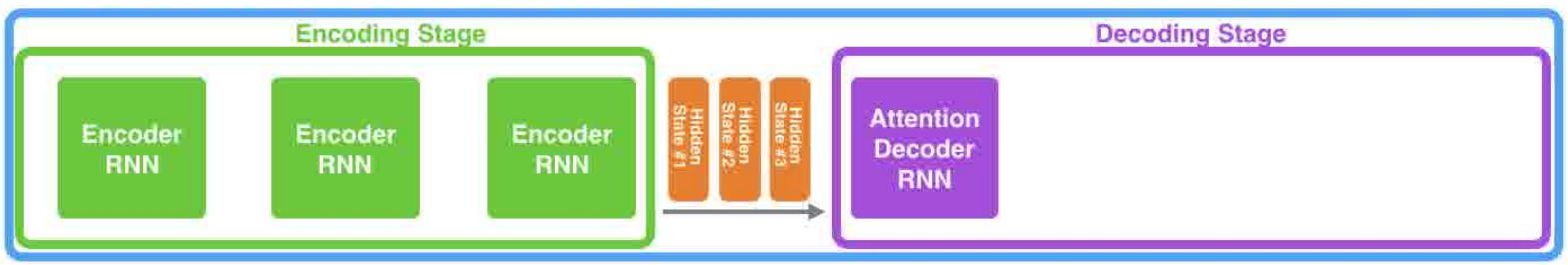
"You can't cram the meaning of a whole %&$# sentence into a single %$&# vector!"

— Ray Mooney

→ Instead of vectors, use matrices!

- fixed number of rows (= size of you hidden state)
- as many columns as words in the input sequence

- How? e.g. concatenate and pass all the hidden layers to the decoder



Recurrent Neural Networks for Neural MT

Sending more information to the decoder is useful, yet we need to also provide it with some **contextual information**...

Convolutional neural networks – (Kalchbrenner et al., 2014)

- Dynamic k-Max Pooling, a global pooling operation over linear sequences to capture context

Bidirectional LSTMs – **(Bahdanau et al., 2015)**

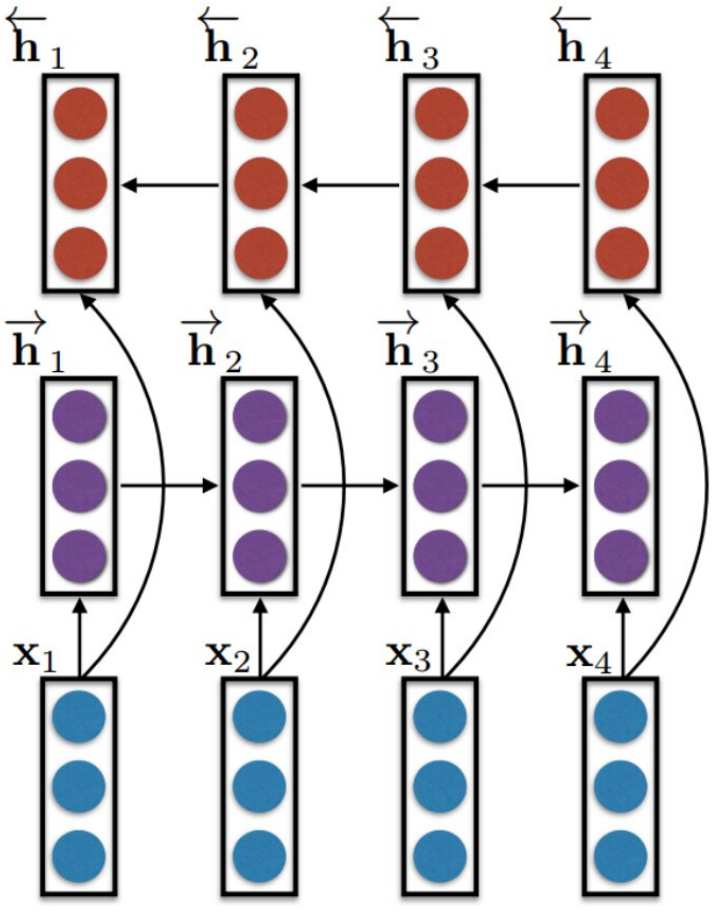
- forward and backward encoding: word representation at step t is the concatenation of its left and right contexts

Transformer model – (Vaswani et al., 2017)

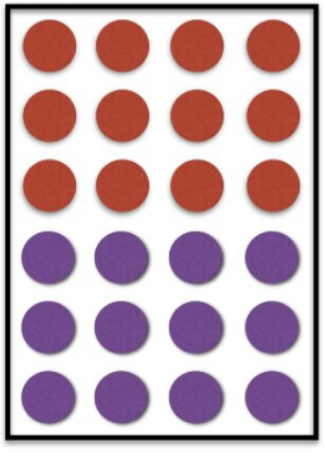
- "Multi-head self-attention"

Bidirectional LSTMs – (Bahdanau et al., 2015)

$$\mathbf{f}_i = [\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{h}}_i]$$



$$\mathbf{F} \in \mathbb{R}^{2n \times |\boldsymbol{f}|}$$



– Question –

How does the decoder know what to translate?

Tip – one of the key principles in MT

– Question –

How does the decoder know what to translate?

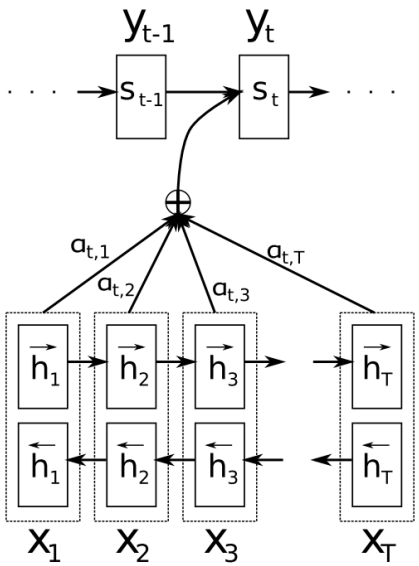
Tip – one of the key principles in MT

(soft) alignment model

Attention mechanism – (Bahdanau et al., 2015)

The decoder now takes as inputs:

- vector of the previous output
- a **weighted** "view" of the input matrix



The weighting of the input is recalculated at **each time-step** (a_t), this is the "**attention distribution**".

Attention mechanism

Motivation: compute the **association** between

- the **previous** decoder state s_{i-1}
  - contains all the information about where we are in the production of the output sequence
- **each** input word $h_j = (\overrightarrow{h_j}, \overleftarrow{h_j})$
  - how relevant each input word is to produce the next output word (soft alignment)

Attention mechanism

The association is computed with a feedforward layer at each time-step, (w, u are vector weights and b, a bias value):

$$a(s_{i-1}, h_j) = w^{aT} s_{i-1} + u^{aT} h_j + b^a$$

Normalisation with softmax:

$$\alpha_{ij} = \frac{\exp(a(s_{i-1}, h_j))}{\sum_k \exp(a(s_{i-1}, h_k))}$$

Creation of a context vector as the **weighted contribution** of all input words:

$$c_i = \sum \alpha_{ij} hj$$

Attention mechanism

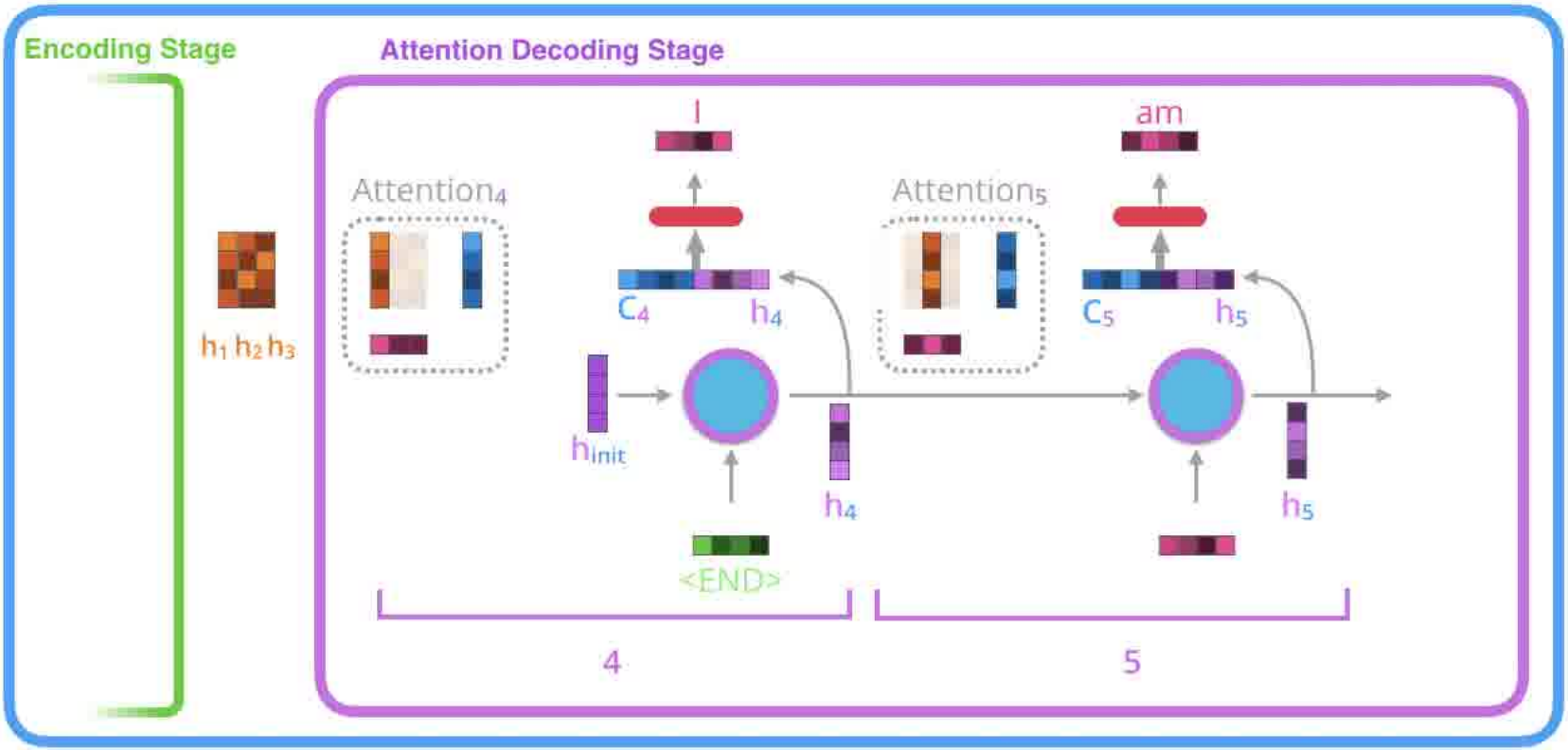


Illustration of a decoding stage with attention.

Attention mechanism

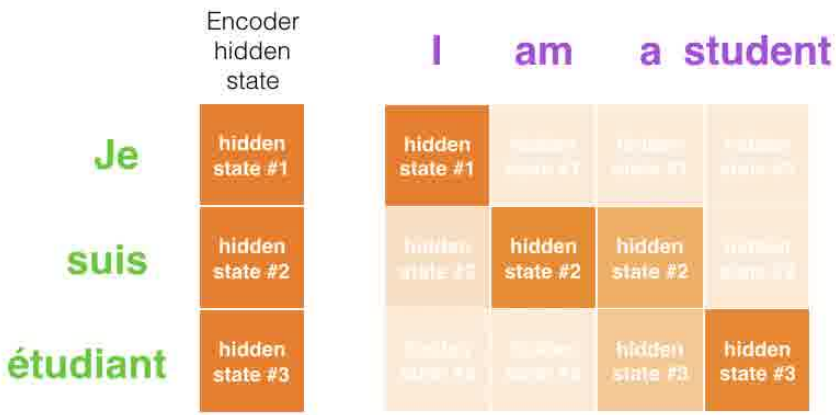
The Attention mechanism...

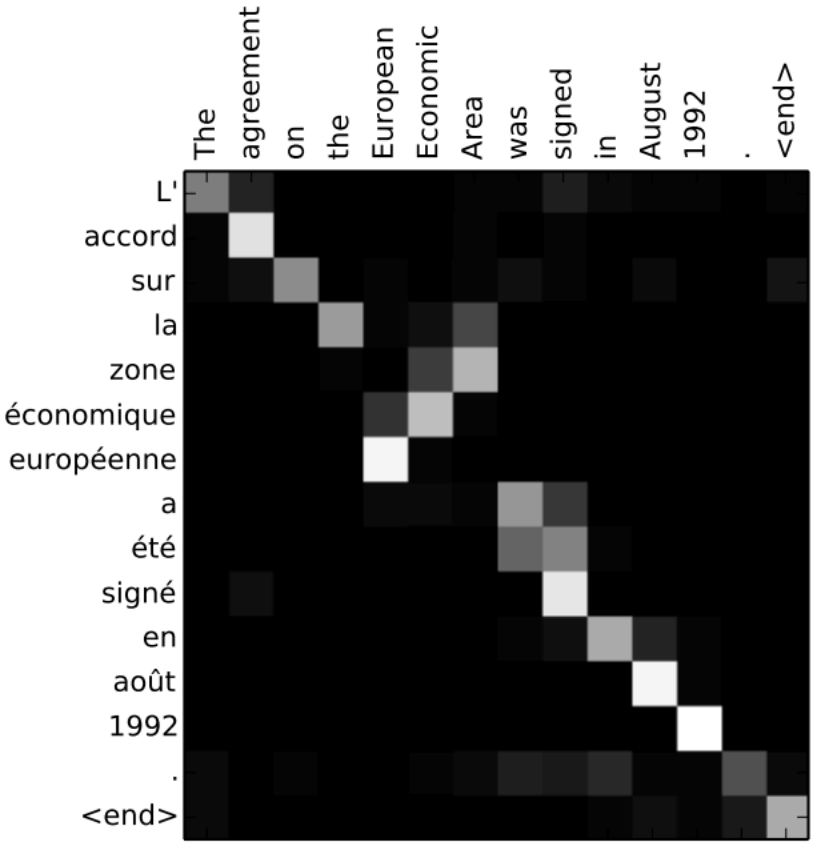
- allows the decoder to focus on the most relevant words of the source sentence
- helps with vanishing gradient problem (provides shortcut to faraway states)
- provides some **interpretability**!

Bonus: the network learns it by itself, we don't have to train an word aligner (compared to Statistical MT!)

Attention mechanism

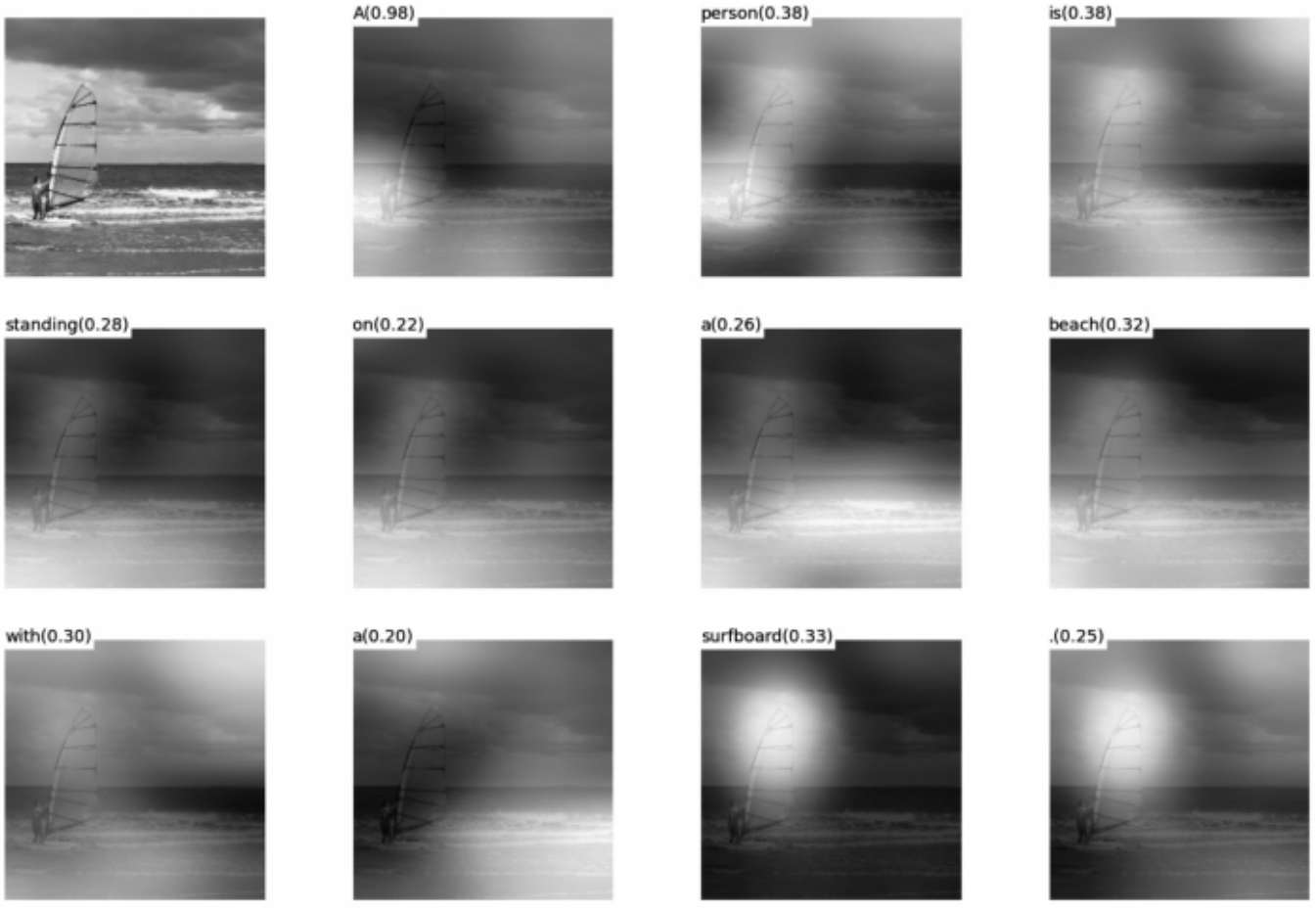
Examples of interpretability





Attention mechanism

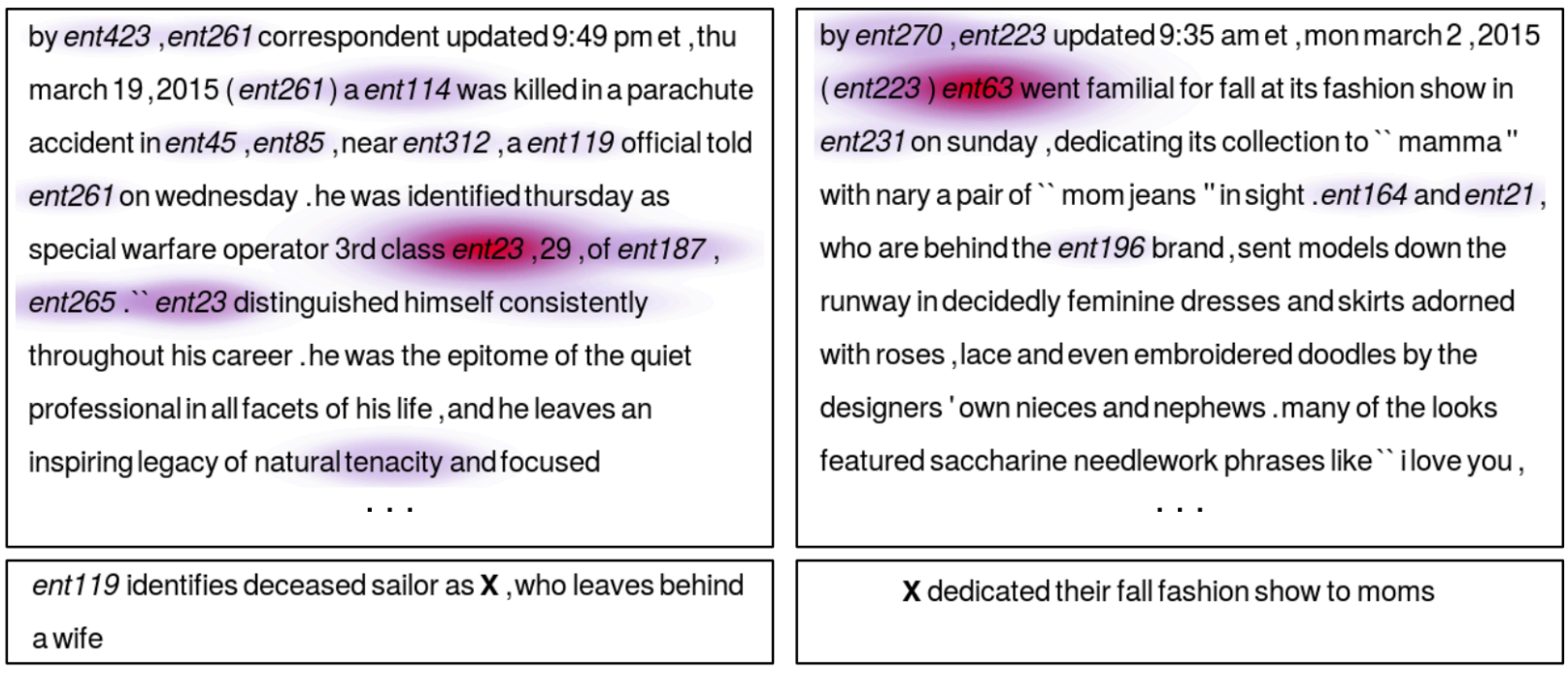
Another example: generation of image caption



(b) A person is standing on a beach with a surfboard.

Attention mechanism

Another example: reading comprehension (question answering)

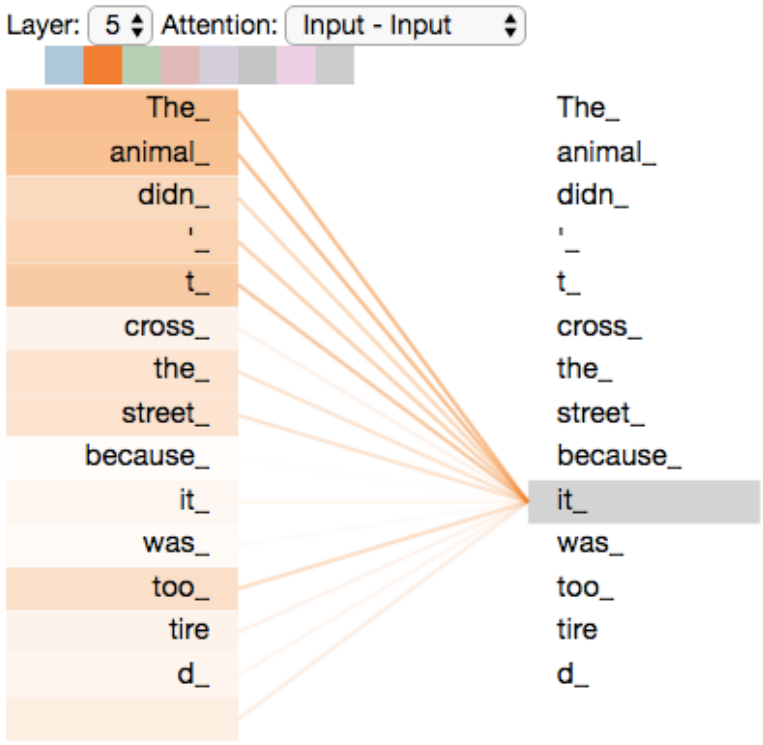


Self-Attention

Self-Attention

→ Refinement: extend Attention to the encoder, by computing the association between any input word and any other input word

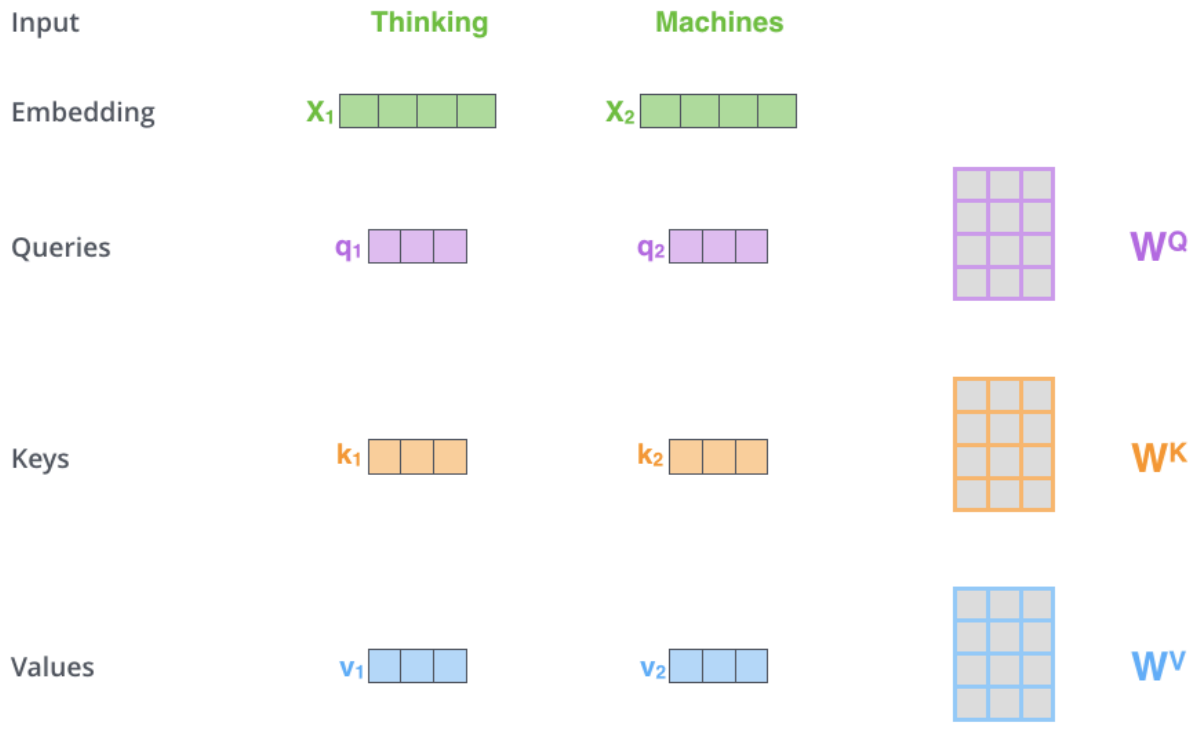
→ refines the representation of each input word by enriching it with context words that help to disambiguate it



Self-Attention – Step 1

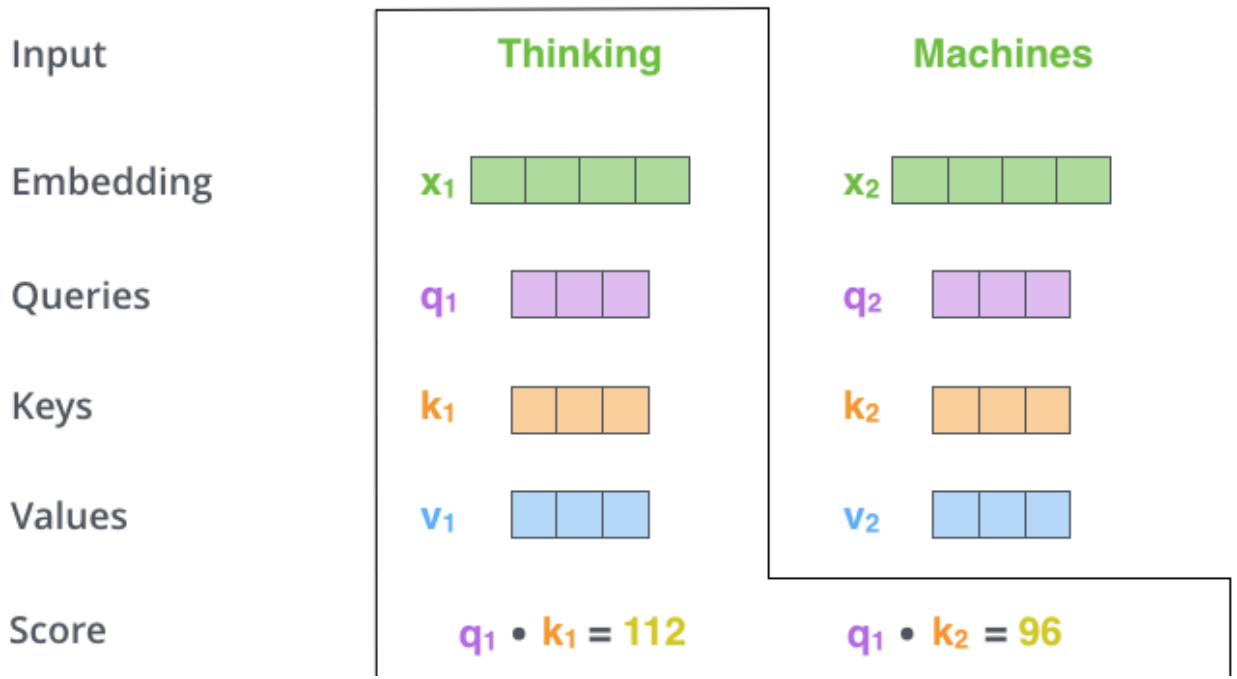
Create three vectors (Query, Key, Value) from each of the encoder's input vectors

- multiply the embedding by three matrices that are trained during the training process



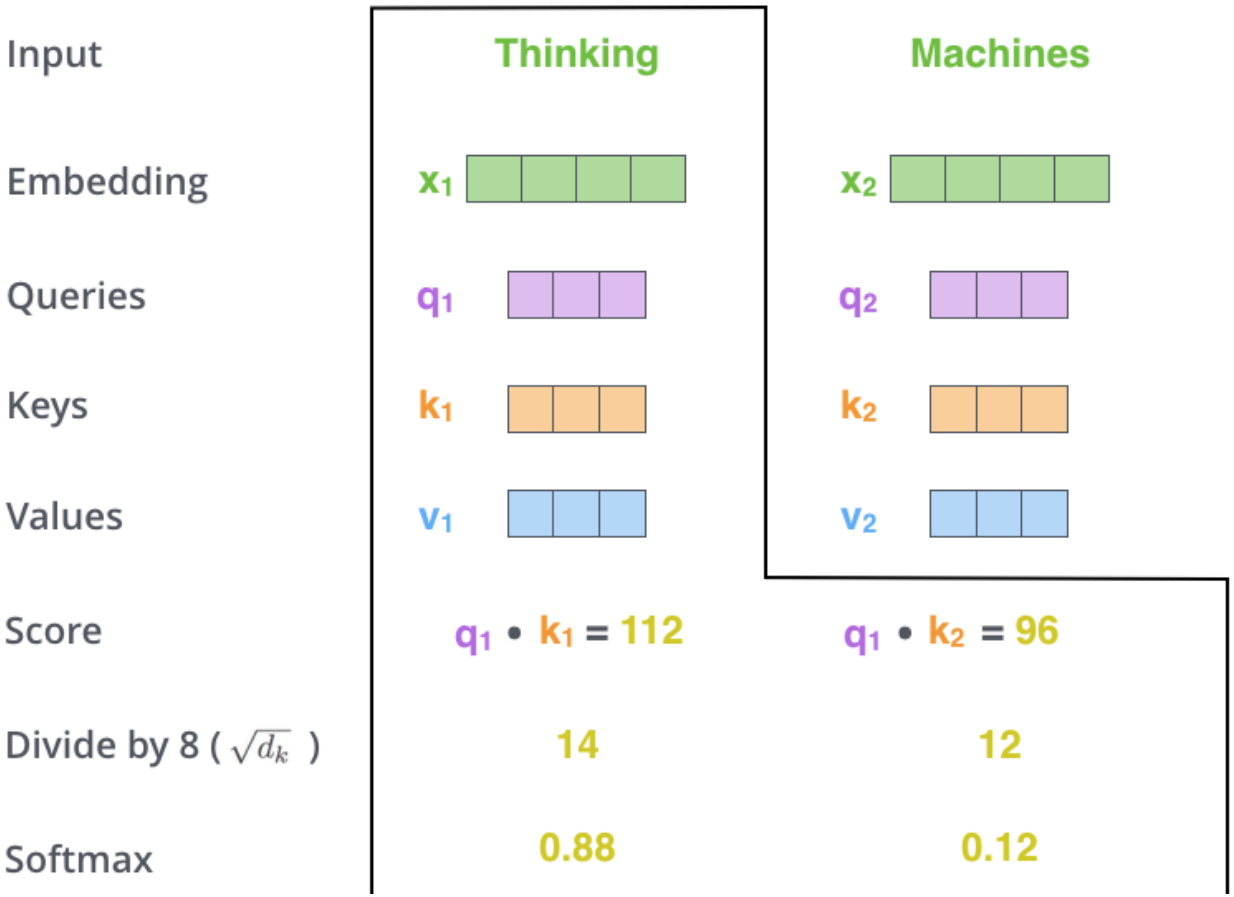
Self-Attention – Step 2

Determine how much focus to place on other parts of the input sentence as we encode a word at a certain position



Self-Attention – Steps 3 & 4

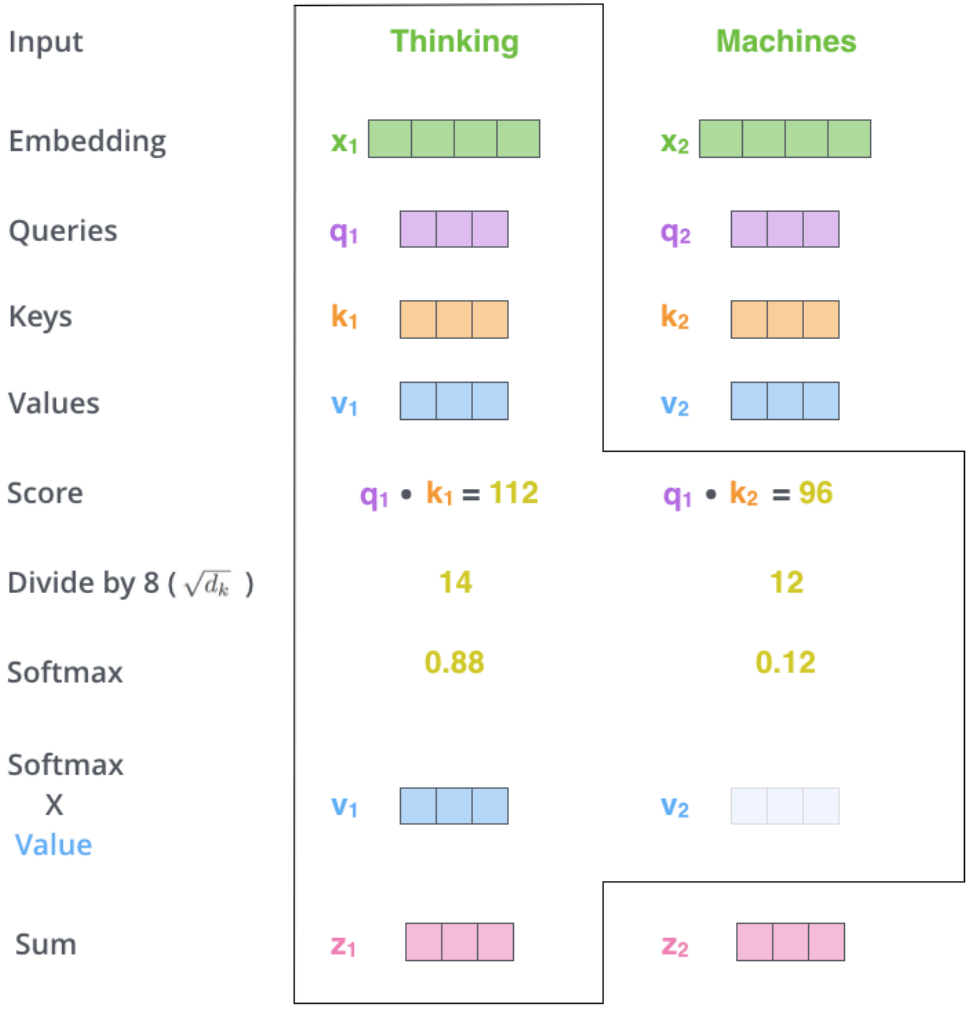
Divide the scores by the square root of the dimension of the key vectors + normalize with softmax so they're all positive and add up to 1



Self-Attention – Step 5 & 6

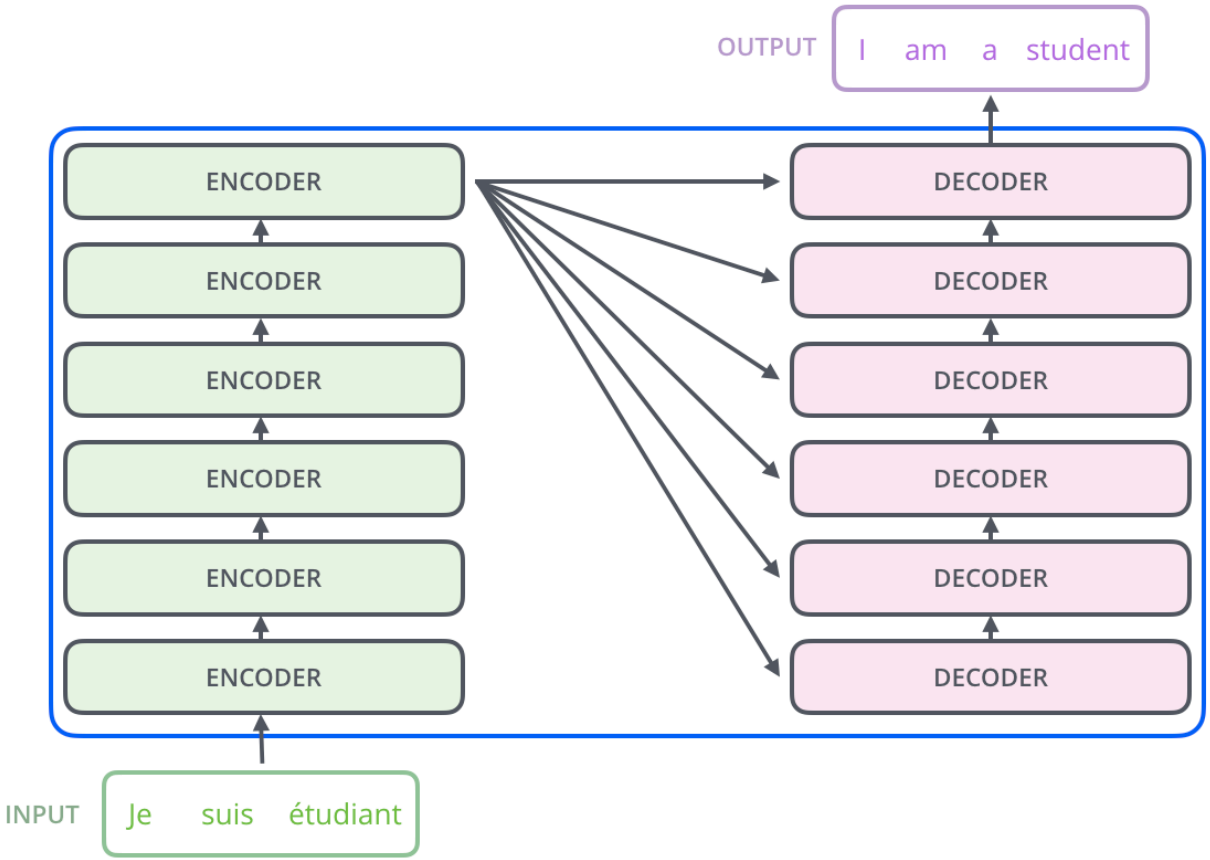
Multiply each value vector by the softmax score, in order to keep the relevant words and drown-out the irrelevant ones

Sum up the weighted value vectors to produce the output of the self-attention layer at this position



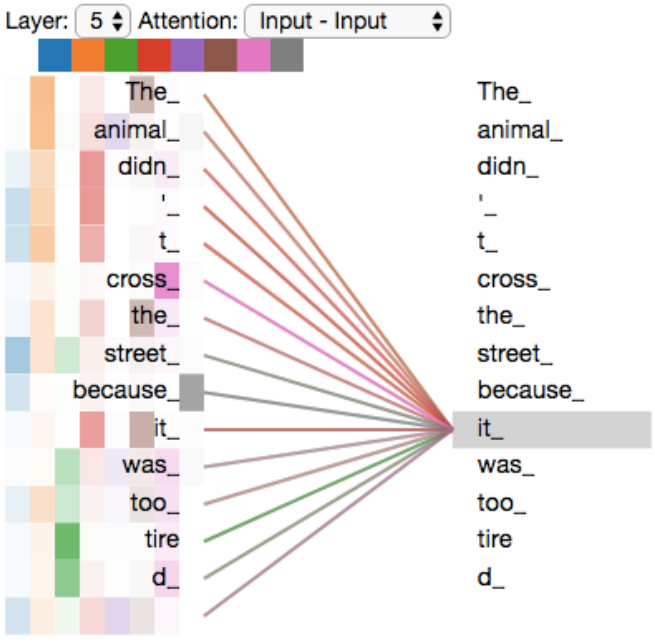
Transformer (Vaswani et al., 2017)

→ Uses Attention to boost the speed with which the model can be trained



Transformer (Vaswani et al., 2017)

Multiple self-attention layers (+ positional embeddings)



→ **harder to optimize** than a RNN sequence-to-sequence model

→ not working out of the box: **hyperparameter tuning** required!

→ current **SOTA**

Machine Translation – Summary

- Rule-based MT
- Example-based MT
- Statistical-based MT
- Neural-based MT with sequence-to-sequence artchitecture (2 RNNs)
  - sequence-to-sequence architecture (2 RNNs)
  - attention and self-attention mechanism

Bibliography

Sources and Inspiration:

- **"Statistical Machine Translation", *by* Philipp Koehn.**
- **"A history of machine translation from the Cold War to deep learning", *by* Ilya Pestov.**
- **"Comparing Neural MT, SMT and RBMT – The SYSTRAN Perspective", *by* Kirti Vashee.**
- **"Machine Translation and Sequence–to-Sequnce Models", *by* André Martins**
- **"Statistical Machine Translation (Chap 13 – Neural MT)", *by* Philipp Koehn**

Bibliography

Sequence to Sequence Deep Learning (Quoc Le, Google)

Stanford CS224N: Translation, Seq2Seq, Attention

Illustrations from `https://jalammar.github.io/`

Original paper on Attention: "Neural Machine Translation by Jointly Learning to Align and Translate" (Bahdanau et al., 2015)

Interesting blog post by Denny Britz: "Attention and Memory in Deep Learning and NLP" (+ other high-quality materials on his blog!)

Appendix

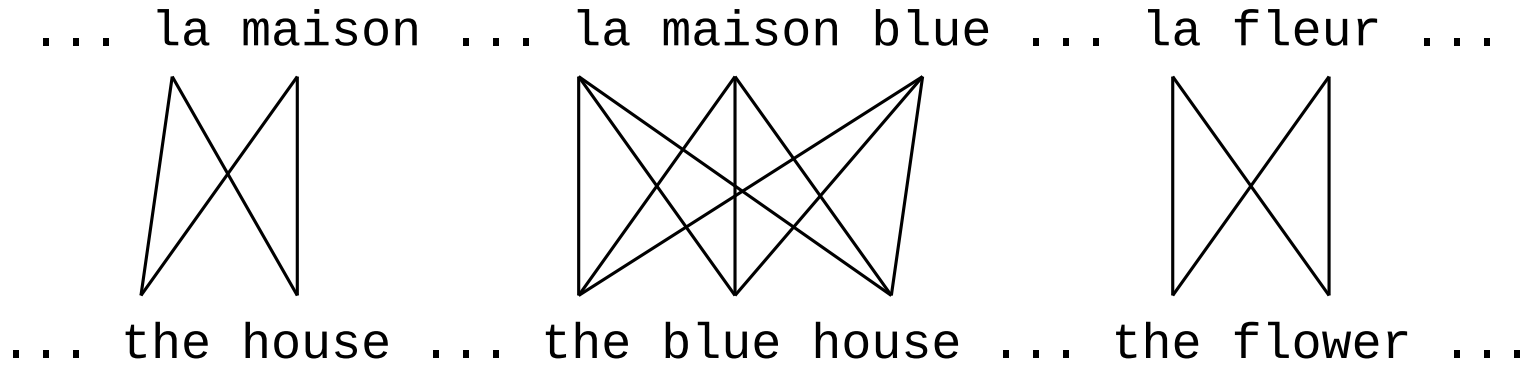
Expectation Maximization (EM) Algorithm

This algorithm is an iterative learning method, that addresses the situation of **incomplete data**:

- if we had *complete data*, we could estimate our *model*
- if we had our *model*, we could fill in the *gaps in the data* (here, to find the most likely alignments between words)

- EM in a nutshell
  1. initialize the model, typically with uniform distributions;
  2. assign probabilities to the missing data;
  3. estimate model parameters from completed data;
  4. iterate steps 2–3 until convergence.

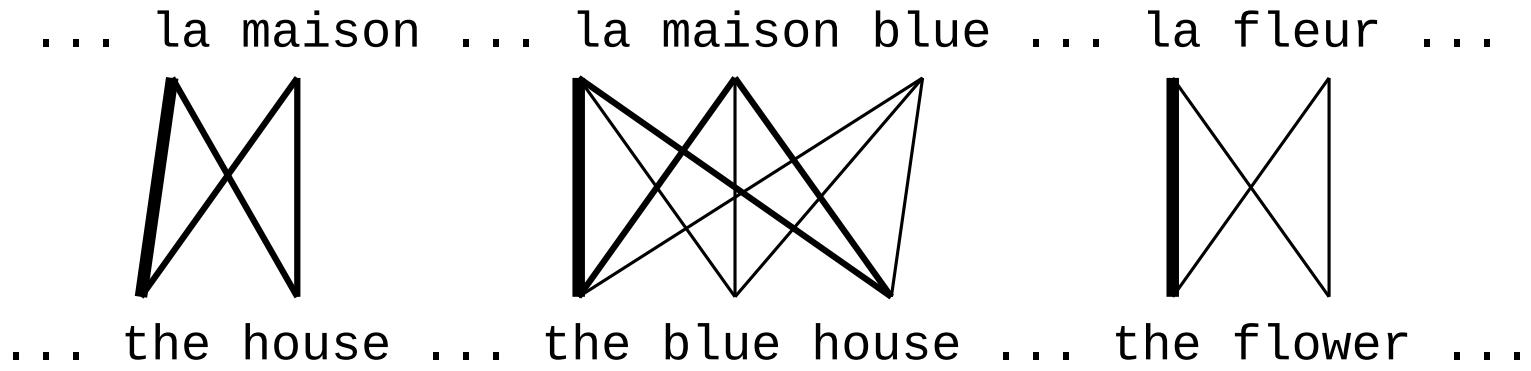
Expectation Maximization (EM) Algorithm



- initial step: all alignments are equally likely

What could our model learn here?

Expectation Maximization (EM) Algorithm

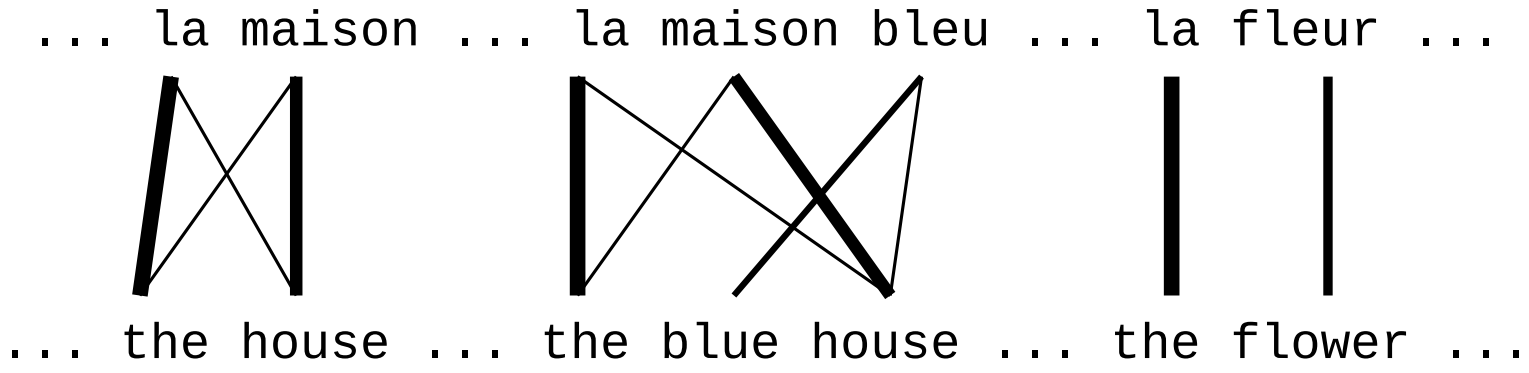


- initial step: all alignments equally likely

What could our model learn here?

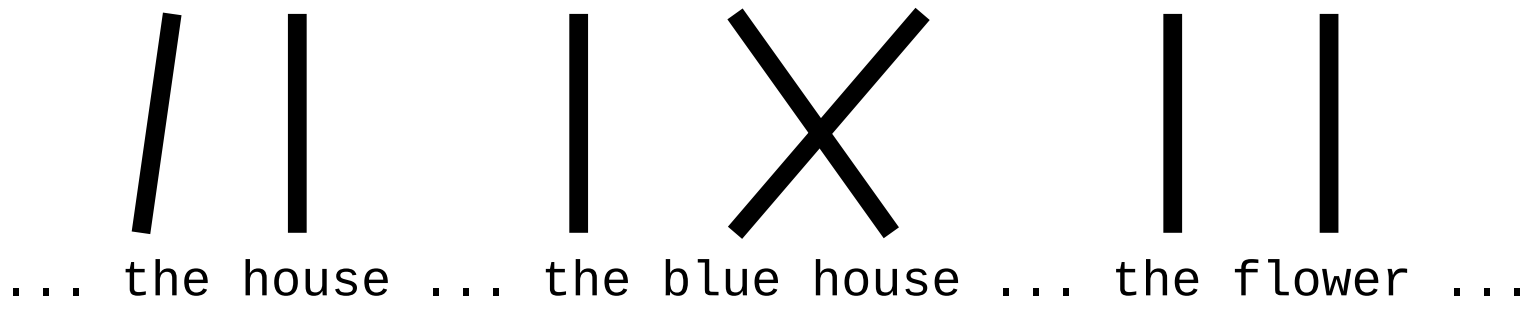
- model learns that, e.g., la is often aligned with the

Expectation Maximization (EM) Algorithm



- After another iteration, it becomes apparent that some alignments, (e.g., between fleur and flower) are more likely (pigeon hole principle).

Expectation Maximization (EM) Algorithm



- Convergence... the inherent hidden structure (alignments), is revealed!

- Parameter estimation from the aligned corpus:

$$
\begin{aligned}
p(\text{la} \mid \text{the}) &= 0.453 \\
p(\text{le} \mid \text{the}) &= 0.334 \\
p(\text{maison} \mid \text{house}) &= 0.876 \\
p(\text{bleu} \mid \text{blue}) &= 0.563 \\
&\ldots
\end{aligned}
$$

EM for IBM Model 1

Similarly, applying the EM algorithm to IBM Model 1, consists of two steps:

- **Expectation**-step: apply our model to the data
  - parts of the model are hidden (i.e. alignments)
  - using the model, assign probabilities to possible alignments

  → we need to compute $p(a|e,f)$

- **Maximization**-step: learn our model from the data
  - take assigned values as fact
  - estimate model from counts

  → we need to collect counts (weighted by probabilities)

- Iterate these steps until convergence

EM for IBM Model 1 – Expectation Step

We need to compute $p(a|\mathbf{e}, \mathbf{f})$, the probability of different alignments given a sentence pair $(\mathbf{e}, \mathbf{f})$:

- Applying Bayes rule:

$$p(a|\mathbf{e}, \mathbf{f}) = \frac{p(\mathbf{e}, a|\mathbf{f})}{p(\mathbf{e}|\mathbf{f})}$$

- We already have the formula for $p(\mathbf{e}, a|\mathbf{f})$ (i.e. probability of translating $\mathbf{f}$ into $\mathbf{e}$ given an alignment a), from the definition of IBM Model 1

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

EM for IBM Model 1 – Expectation Step

We still need to derive $p(\mathbf{e}|\mathbf{f})$, the probability of translating the sentence $\mathbf{f}$ into $\mathbf{e}$, with **any** alignment:

$$
\begin{aligned}
p(\mathbf{e}|\mathbf{f}) &= \sum_{a} p(\mathbf{e}, a|\mathbf{f}) \\
&= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} p(\mathbf{e}, a|\mathbf{f}) \\
&= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) \\
&= \frac{\epsilon}{(l_f+1)^{l_e}} \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) \\
&= \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)
\end{aligned}
$$

EM for IBM Model 1 – Expectation Step

BONUS – short example with $l_e = l_f = 2$:

$$\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} = \frac{\epsilon}{3^2} \prod_{j=1}^{2} t(e_j|f_{a(j)}) =$$

$$\begin{aligned}
&= t(e_1|f_0)\ t(e_2|f_0) + t(e_1|f_0)\ t(e_2|f_1) + t(e_1|f_0)\ t(e_2|f_2) + \\
&\quad + t(e_1|f_1)\ t(e_2|f_0) + t(e_1|f_1)\ t(e_2|f_1) + t(e_1|f_1)\ t(e_2|f_2) + \\
&\quad + t(e_1|f_2)\ t(e_2|f_0) + t(e_1|f_2)\ t(e_2|f_1) + t(e_1|f_2)\ t(e_2|f_2) = \\
&= t(e_1|f_0)\,(t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) + \\
&\quad + t(e_1|f_1)\,(t(e_2|f_1) + t(e_2|f_1) + t(e_2|f_2)) + \\
&\quad + t(e_1|f_2)\,(t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2)) \\
&= (t(e_1|f_0) + t(e_1|f_1) + t(e_1|f_2))\,(t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2))
\end{aligned}$$

EM for IBM Model 1 – Expectation Step

Putting what we have together:

$$p(\mathrm{a}|\mathbf{e},\mathbf{f}) = p(\mathbf{e},\mathrm{a}|\mathbf{f})/p(\mathbf{e}|\mathbf{f})$$

$$= \frac{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})}{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)}$$

$$= \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)}$$

EM for IBM Model 1 – Maximisation Step

Now we have to collect counts, by collecting evidence from a sentence pair (**f**, **e**), that a particular input word f translates into the output word e:

$$c(e|f;\mathbf{e},\mathbf{f}) = \sum_a p(a|\mathbf{e},\mathbf{f}) \sum_{j=1}^{l_e} \delta(e,e_j)\delta(f,f_{a(j)})$$

$$= \frac{t(e|f)}{\sum_{i=0}^{l_f} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e,e_j) \sum_{i=0}^{l_f} \delta(f,f_i)$$

The Kronecker function $\delta(x,y)$ is 1 if $x=y$, 0 otherwise.

For every co-occurence in a sentence pair, we add the probability $t(e|f)$ to the count.

EM for IBM Model 1 – Maximisation Step

Finally, by **normalizing** these counts, we can estimate the new translation probability distribution:

$$t(e|f;\mathbf{e},\mathbf{f}) = \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e|f;\mathbf{e},\mathbf{f}))}{\sum_{e}\sum_{(\mathbf{e},\mathbf{f})} c(e|f;\mathbf{e},\mathbf{f}))}$$