Machine Translation

Natural Language Processing (COM4513/6513)

Fred. Blain

Lecture 16 — April 27th, 2018

Department of Computer Science
University of Sheffield
f.blain@sheffield.ac.uk

[...] 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." [...]

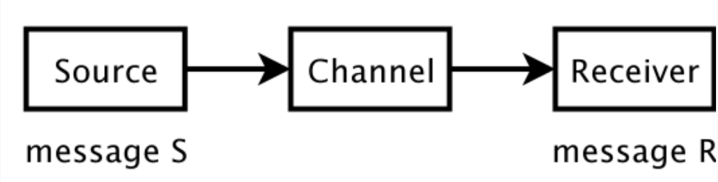
— W. Weaver

[...] 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." [...]

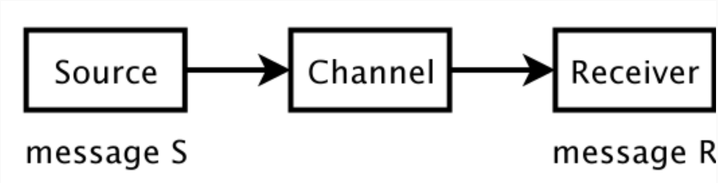
— W. Weaver (1947)

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)

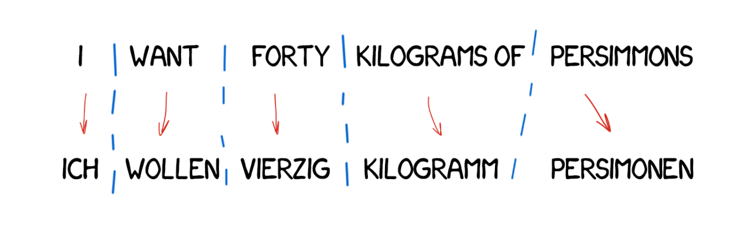
Noisy Channel Model

Machine Translation

Generative process: breaking up translation process into smaller steps

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

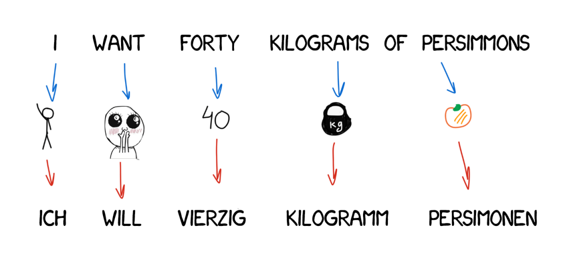
→ translating a word = assigning a word in target language



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)

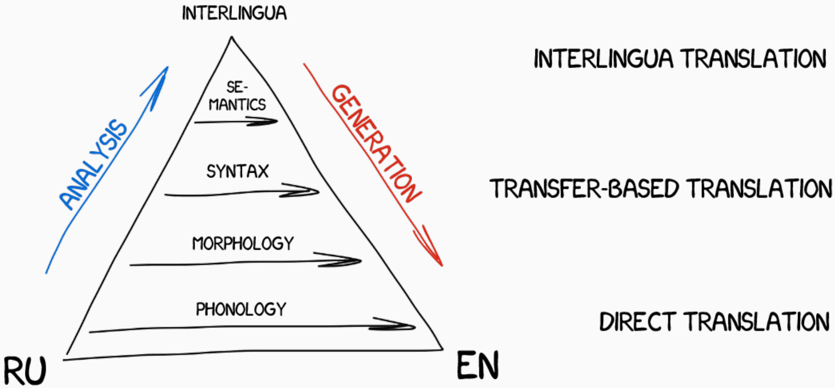


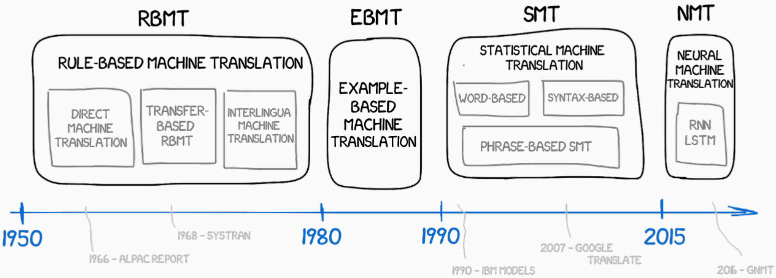
Fig. The Vauquois Triangle

"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

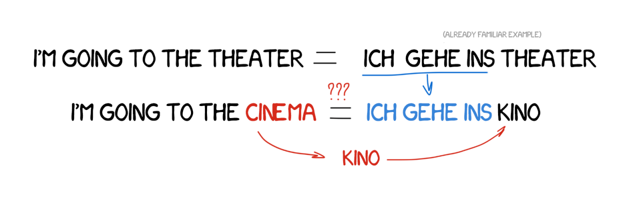
Timeline of Machine Translation



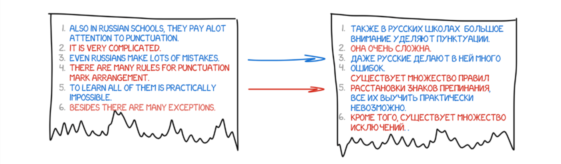
Data-driven methods (1980-201?)

→ Learn how to translate from past translation examples!

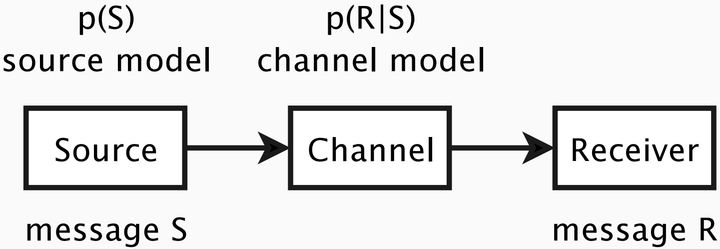
- **EXAMPLE**-based Machine Translation (EBMT)



- **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:

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

- $p(\mathbf{e})$: 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 #2: "Language modeling basics"
→ Cf. Lecture #4: "Part-of-speech tagging with HMMs (and Viterbi)"

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 –

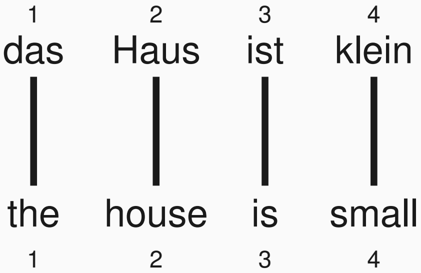
How do we obtain those counts?

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

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

– 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

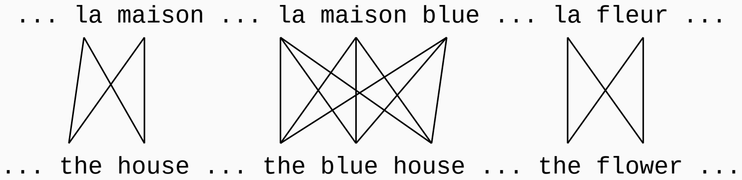
Expectation Maximization (EM) Algorithm

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

- if we had *complete data*, would 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.

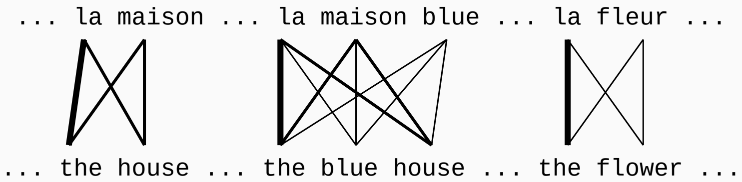
Expectation Maximization (EM) Algorithm



- initial step: all alignments are equally likely

What could our model learn here?

Expectation Maximization (EM) Algorithm

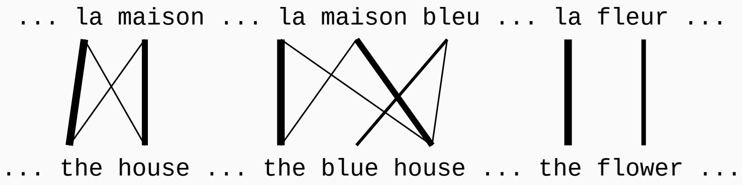


- initial step: all alignments equally likely

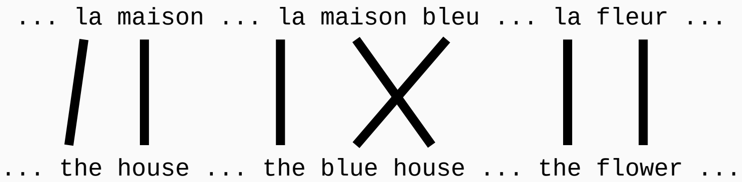
What could our model learn here?

- 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).



- Convergence… the inherent hidden structure (alignments), is revealed!
- Parameter estimation from the aligned corpus:

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

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

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

EM for IBM Model 1 – Expectation Step

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

$$
\begin{aligned}
\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} &= \frac{\epsilon}{3^2} \prod_{j=1}^{2} t(e_j|f_{a(j)}) = \\
&= t(e_1|f_0)\ t(e_2|f_0) + t(e_1|f_0)\ t(e_2|f_1) + t(e_1|f_0)\ t(e_2|f_2) + \\
&\quad + t(e_1|f_1)\ t(e_2|f_0) + t(e_1|f_1)\ t(e_2|f_1) + t(e_1|f_1)\ t(e_2|f_2) + \\
&\quad + t(e_1|f_2)\ t(e_2|f_0) + t(e_1|f_2)\ t(e_2|f_1) + t(e_1|f_2)\ t(e_2|f_2) = \\
&= t(e_1|f_0)\,(t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) + \\
&\quad + t(e_1|f_1)\,(t(e_2|f_1) + t(e_2|f_1) + t(e_2|f_2)) + \\
&\quad + t(e_1|f_2)\,(t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2)) \\
&= \ (t(e_1|f_0) + t(e_1|f_1) + t(e_1|f_2))\,(t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2))
\end{aligned}
$$

EM for IBM Model 1 – Expectation Step

Putting what we have together:

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

EM for IBM Model 1 – Maximisation Step

Now we have to collect counts, by collecting evidence from a sentence pair $\mathbf{f}$ into $\mathbf{e}$, 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.

EM for IBM Model 1 – Maximisation Step

After collecting these counts over a parallel corpus, we can estimate the new translation probability distribution:

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

– Question –

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

- Sometime one is preferred over the other
  - small step: 357,000,000 occurrences in the Google index
  - little step: 34,000,000 occurrences in the Google index

– Question –

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

- Sometime one is preferred over the other
  - small step: 357,000,000 occurrences in the Google index
  - little step: 34,000,000 occurrences in the Google index

Language Model

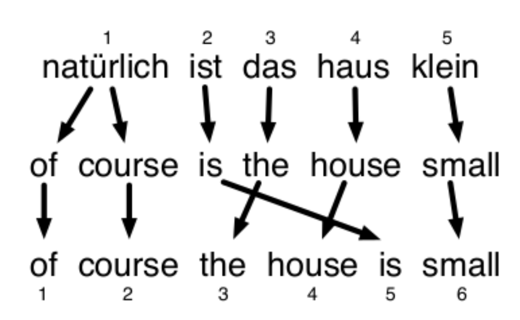
– Question –

What are the flaws of the IBM Model 1?

IBM Model 2

Considering the word order in sentences

- 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:

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

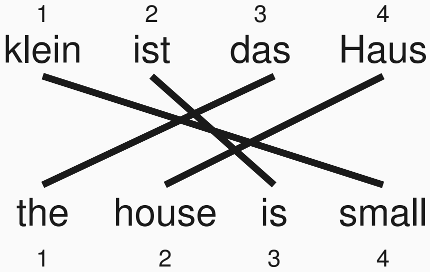
- Putting everything together

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

EM training of this model works the same way as IBM Model 1

Reordering

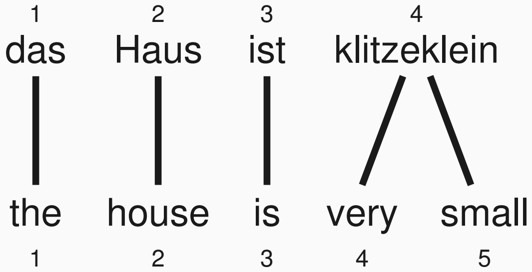
IBM Model 1 considers all possible reordering as equally likely. Often, words that follows each other in one language have translations that follow each other in the output language



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

One-to-Many Translation

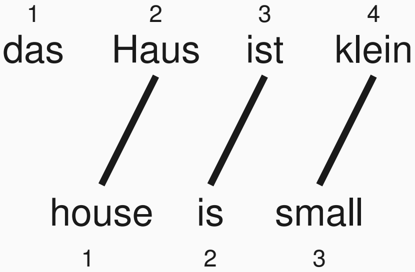
A source word may translate into multiple target words (a.k.a **fertility**)



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

Dropping Words

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

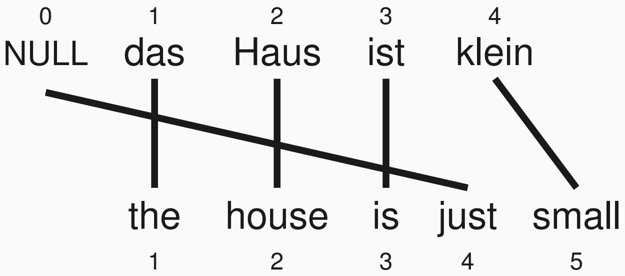


$$a : \{1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4\}$$

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



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

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 reordering model
IBM Model 5	fixes deficiency

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

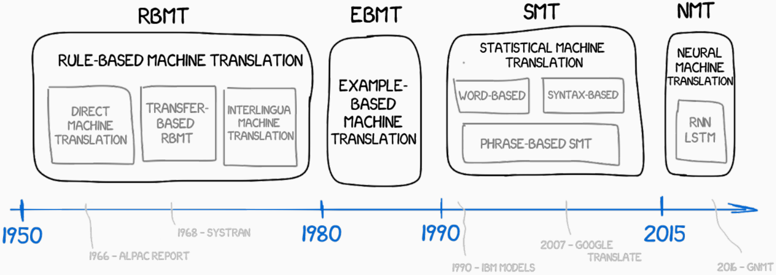
Does not constitute the state-of-the-art anymore, but <u>many of the principles and methods are still current today!</u>

”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 [...]”

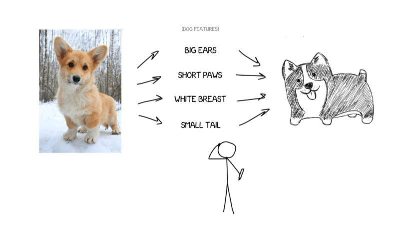
— J. Senellart (2016)

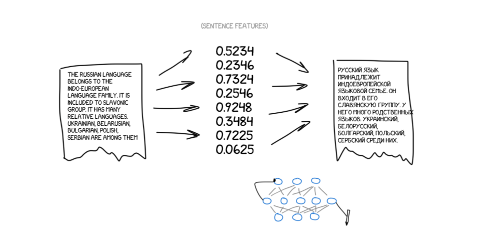
Global CTO at SYSTRAN

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)





In NMT...

- one neural network only encodes the sentence to the specific set of features;
- another one only decodes them back to the text.

→ Both have no idea about the each other, and each of them knows only its own language: Interlingua is back!

Current State-Of-The-Art in Machine Translation!

Summary

1. Rule-Based Machine Translation
2. Data-driven methods (1980-201?)
3. Lexical Translation Probabilities
4. Neural Machine Translation

Bibliography

”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.

`f.blain@sheffield.ac.uk`