

---

# IBM Model 1 and the EM Algorithm

Philipp Koehn

10 September 2020



# Lexical Translation



- How to translate a word → look up in dictionary
  - 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**
- Note: In all lectures, we translate from a foreign language into English

# Collect Statistics



Look at a parallel corpus (German text along with English translation)

<b>Translation of <i>Haus</i></b>	<b>Count</b>
house	8,000
building	1,600
home	200
household	150
shell	50

# Estimate Translation Probabilities



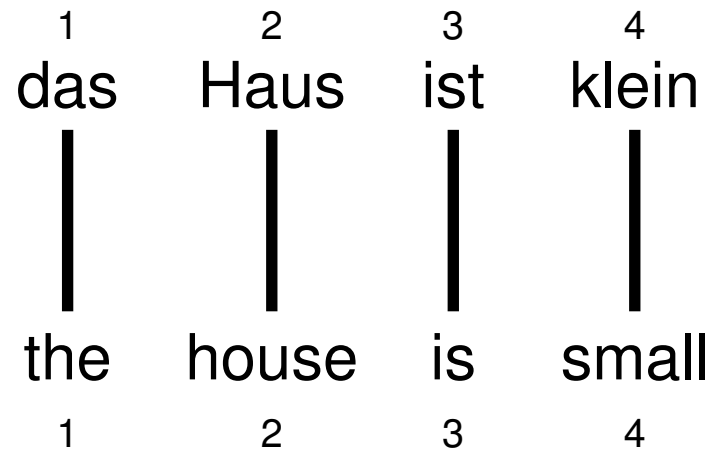
Maximum likelihood estimation

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

# Alignment



- In a parallel text (or when we translate), we align words in one language with the words in the other



- Word positions are numbered 1–4

# Alignment Function



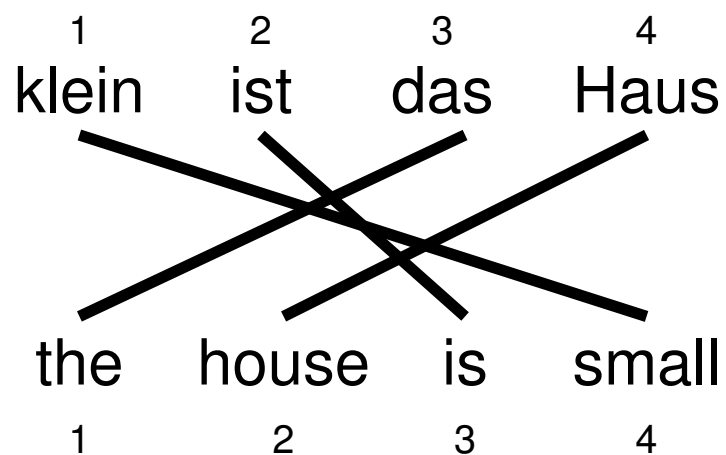
- Formalizing alignment with an alignment function
- Mapping an English target word at position  $i$  to a German source word at position  $j$  with a function  $a : i \rightarrow j$
- Example

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

# Reordering



Words may be reordered during translation

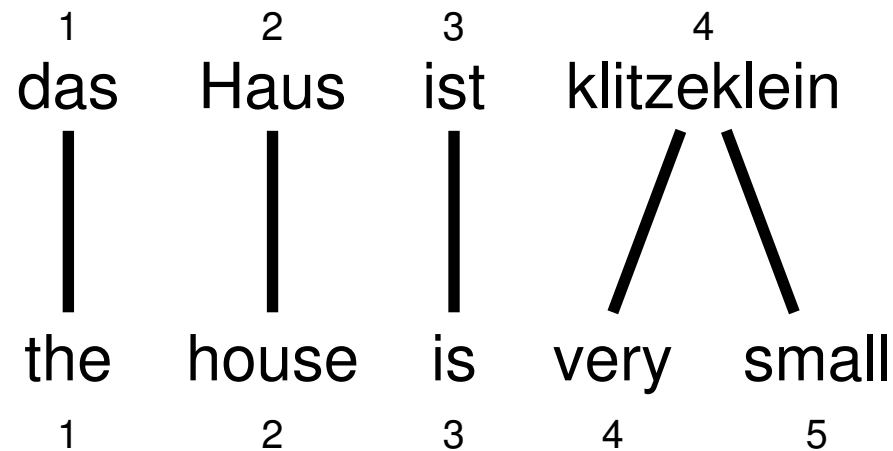


$$a : \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\}$$

# One-to-Many Translation



A source word may translate into multiple target words

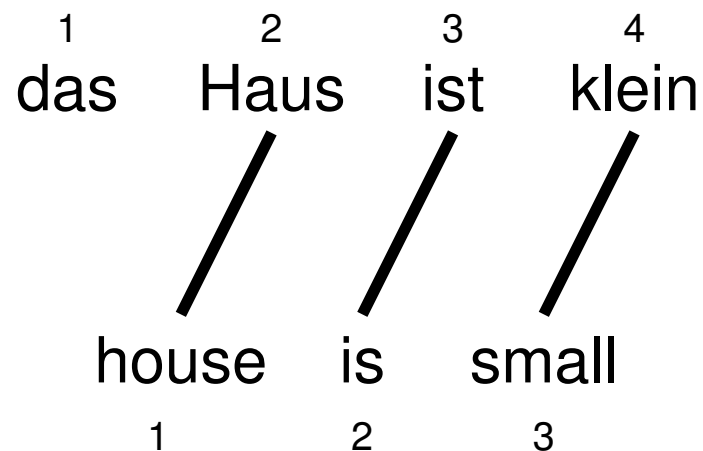


$$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4, 5 \rightarrow 4\}$$



# Dropping Words

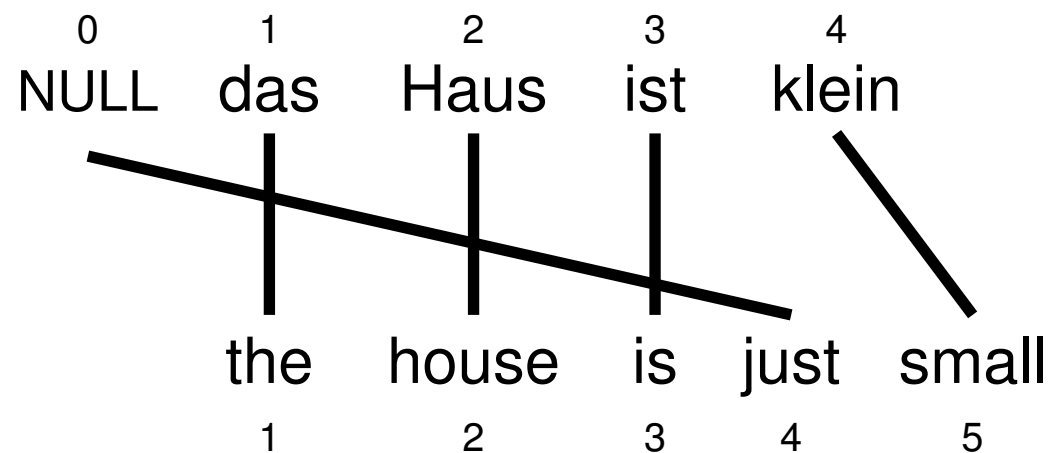
Words may be dropped when translated  
(German article **das** is dropped)



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

# 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



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

# IBM Model 1

- Generative model: break up translation process into smaller steps
  - IBM Model 1 only uses lexical translation
- Translation probability
  - for a foreign sentence  $\mathbf{f} = (f_1, \dots, f_{l_f})$  of length  $l_f$
  - to an English sentence  $\mathbf{e} = (e_1, \dots, e_{l_e})$  of length  $l_e$
  - with an alignment of each English word  $e_j$  to a foreign word  $f_i$  according to the alignment function  $a : j \rightarrow i$

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

- parameter  $\epsilon$  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

$$\begin{aligned} p(e, a|f) &= \frac{\epsilon}{4^3} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein}) \\ &= \frac{\epsilon}{4^3} \times 0.7 \times 0.8 \times 0.8 \times 0.4 \\ &= 0.0028\epsilon \end{aligned}$$

# finding translations

# Centauri-Arcturan Parallel Text

1a. ok-voon ororok sprok .  
1b. at-voon bichat dat .

---

2a. ok-drubel ok-voon anak plok sprok .  
2b. at-drubel at-voon pippat rrat dat .

---

3a. erok sprok izok hihok ghirok .  
3b. totat dat arrat vat hilat .

---

4a. ok-voon anak drok brok jok .  
4b. at-voon krat pippat sat lat .

---

5a. wiwok farok izok stok .  
5b. totat jjat quat cat .

---

6a. lalok sprok izok jok stok .  
6b. wat dat krat quat cat .

---

7a. lalok farok ororok lalok sprok izok enemok .  
7b. wat jjat bichat wat dat vat eneat .

---

8a. lalok brok anak plok nok .  
8b. iat lat pippat rrat nnat .

---

9a. wiwok nok izok kantok ok-yurp .  
9b. totat nnat quat oloat at-yurp .

---

10a. lalok mok nok yorok ghirok klok .  
10b. wat nnat gat mat bat hilat .

---

11a. lalok nok crrrok hihok yorok zanzanok .  
11b. wat nnat arrat mat zanzanat .

---

12a. lalok rarok nok izok hihok mok .  
12b. wat nnat forat arrat vat gat .

---

Translation challenge: **farok crrrok hihok yorok klok kantok ok-yurp**

(from Knight (1997): Automating Knowledge Acquisition for Machine Translation)

# em algorithm

# Learning Lexical Translation Models



- We would like to estimate the lexical translation probabilities  $t(e|f)$  from a parallel corpus
- ... but we do not have the alignments
- 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*

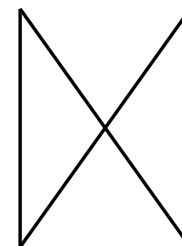
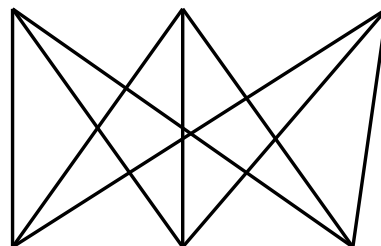
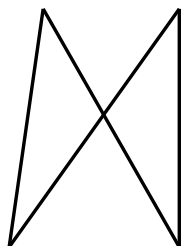


# EM Algorithm

- Incomplete data
  - if we had *complete data*, would could estimate *model*
  - if we had *model*, we could fill in the *gaps in the data*
- Expectation Maximization (EM) in a nutshell
  1. initialize model parameters (e.g. uniform)
  2. assign probabilities to the missing data
  3. estimate model parameters from completed data
  4. iterate steps 2–3 until convergence

# EM Algorithm

... la maison ... la maison blue ... la fleur ...

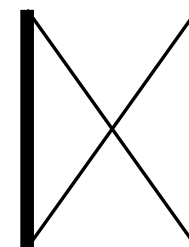
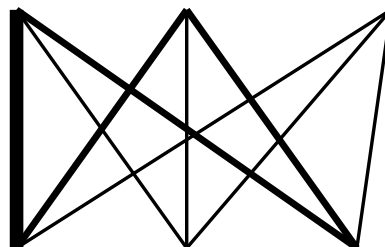
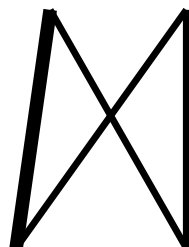


... the house ... the blue house ... the flower ...

- Initial step: all alignments equally likely
- Model learns that, e.g., **la** is often aligned with **the**

# EM Algorithm

... la maison ... la maison blue ... la fleur ...

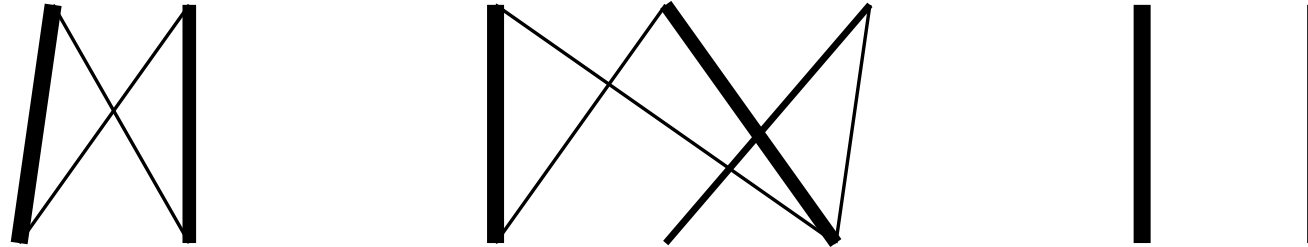


... the house ... the blue house ... the flower ...

- After one iteration
- Alignments, e.g., between **la** and **the** are more likely

# EM Algorithm

... la maison ... la maison bleu ... la fleur ...



... the house ... the blue house ... the flower ...

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

# EM Algorithm

... la maison ... la maison bleu ... la fleur ...  
/ | | X | |  
... the house ... the blue house ... the flower ...

- Convergence
- Inherent hidden structure revealed by EM

# EM Algorithm

... la maison ... la maison bleu ... la fleur ...  
/ | | X | |  
... the house ... the blue house ... the flower ...



$p(\text{la}|\text{the}) = 0.453$   
 $p(\text{le}|\text{the}) = 0.334$   
 $p(\text{maison}|\text{house}) = 0.876$   
 $p(\text{bleu}|\text{blue}) = 0.563$   
...

- Parameter estimation from the aligned corpus

# IBM Model 1 and EM

- EM Algorithm consists of two steps
- Expectation-Step: Apply model to the data
  - parts of the model are hidden (here: alignments)
  - using the model, assign probabilities to possible values
- Maximization-Step: Estimate model from data
  - take assign values as fact
  - collect counts (weighted by probabilities)
  - estimate model from counts
- Iterate these steps until convergence

# IBM Model 1 and EM

- We need to be able to compute:
  - Expectation-Step: probability of alignments
  - Maximization-Step: count collection



# IBM Model 1 and EM

- **Probabilities**

$$p(\text{the}|\text{la}) = 0.7 \quad p(\text{house}|\text{la}) = 0.05$$

$$p(\text{the}|\text{maison}) = 0.1 \quad p(\text{house}|\text{maison}) = 0.8$$

- **Alignments**



$$p(\mathbf{e}, a|\mathbf{f}) = 0.56 \quad p(\mathbf{e}, a|\mathbf{f}) = 0.035 \quad p(\mathbf{e}, a|\mathbf{f}) = 0.08 \quad p(\mathbf{e}, a|\mathbf{f}) = 0.005$$

$$p(a|\mathbf{e}, \mathbf{f}) = 0.824 \quad p(a|\mathbf{e}, \mathbf{f}) = 0.052 \quad p(a|\mathbf{e}, \mathbf{f}) = 0.118 \quad p(a|\mathbf{e}, \mathbf{f}) = 0.007$$

- **Counts**

$$c(\text{the}|\text{la}) = 0.824 + 0.052 \quad c(\text{house}|\text{la}) = 0.052 + 0.007$$

$$c(\text{the}|\text{maison}) = 0.118 + 0.007 \quad c(\text{house}|\text{maison}) = 0.824 + 0.118$$

# IBM Model 1 and EM: Expectation Step



- We need to compute  $p(a|\mathbf{e}, \mathbf{f})$
- Applying the chain 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  $p(\mathbf{e}, \mathbf{a}|\mathbf{f})$  (definition of Model 1)

- We need to compute  $p(\mathbf{e}|\mathbf{f})$

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

# IBM Model 1 and EM: Expectation Step

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

- Note the trick in the last line
  - removes the need for an exponential number of products
  - this makes IBM Model 1 estimation tractable

# The Trick

(case  $l_e = l_f = 2$ )

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

- Combine what we have:

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

# IBM Model 1 and EM: Maximization Step

30



- Now we have to collect counts
- Evidence from a sentence pair  $\mathbf{e}, \mathbf{f}$  that word  $e$  is a translation of word  $f$ :

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

- With the same simplification as before:

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



After collecting these counts over a corpus, we can estimate the model:

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



# IBM Model 1 and EM: Pseudocode

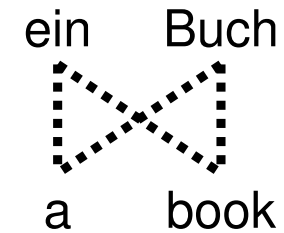
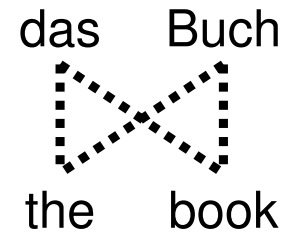
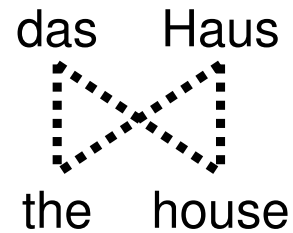
**Input:** set of sentence pairs  $(\mathbf{e}, \mathbf{f})$

**Output:** translation prob.  $t(e|f)$

```
1: initialize  $t(e|f)$  uniformly
2: while not converged do
3:   // initialize
4:    $\text{count}(e|f) = 0$  for all  $e, f$ 
5:    $\text{total}(f) = 0$  for all  $f$ 
6:   for all sentence pairs  $(\mathbf{e}, \mathbf{f})$  do
7:     // compute normalization
8:     for all words  $e$  in  $\mathbf{e}$  do
9:        $\text{s-total}(e) = 0$ 
10:      for all words  $f$  in  $\mathbf{f}$  do
11:         $\text{s-total}(e) += t(e|f)$ 
12:      end for
13:    end for
```

```
14:    // collect counts
15:    for all words  $e$  in  $\mathbf{e}$  do
16:      for all words  $f$  in  $\mathbf{f}$  do
17:         $\text{count}(e|f) += \frac{t(e|f)}{\text{s-total}(e)}$ 
18:         $\text{total}(f) += \frac{t(e|f)}{\text{s-total}(e)}$ 
19:      end for
20:    end for
21:  end for
22:  // estimate probabilities
23:  for all foreign words  $f$  do
24:    for all English words  $e$  do
25:       $t(e|f) = \frac{\text{count}(e|f)}{\text{total}(f)}$ 
26:    end for
27:  end for
28: end while
```

# Convergence



<i>e</i>	<i>f</i>	initial	1st it.	2nd it.	3rd it.	...	final
the	das	0.25	0.5	0.6364	0.7479	...	1
book	das	0.25	0.25	0.1818	0.1208	...	0
house	das	0.25	0.25	0.1818	0.1313	...	0
the	buch	0.25	0.25	0.1818	0.1208	...	0
book	buch	0.25	0.5	0.6364	0.7479	...	1
a	buch	0.25	0.25	0.1818	0.1313	...	0
book	ein	0.25	0.5	0.4286	0.3466	...	0
a	ein	0.25	0.5	0.5714	0.6534	...	1
the	haus	0.25	0.5	0.4286	0.3466	...	0
house	haus	0.25	0.5	0.5714	0.6534	...	1

# Perplexity

- How well does the model fit the data?
- Perplexity: derived from probability of the training data according to the model

$$\log_2 PP = - \sum_s \log_2 p(\mathbf{e}_s | \mathbf{f}_s)$$

- Example ( $\epsilon=1$ )

	initial	1st it.	2nd it.	3rd it.	...	final
$p(\text{the haus}   \text{das haus})$	0.0625	0.1875	0.1905	0.1913	...	0.1875
$p(\text{the book}   \text{das buch})$	0.0625	0.1406	0.1790	0.2075	...	0.25
$p(\text{a book}   \text{ein buch})$	0.0625	0.1875	0.1907	0.1913	...	0.1875
perplexity	4095	202.3	153.6	131.6	...	113.8

# 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

- Only IBM Model 1 has global maximum
  - training of a higher IBM model builds on previous model
- Computationally biggest change in Model 3
  - trick to simplify estimation does not work anymore
  - exhaustive count collection becomes computationally too expensive
  - sampling over high probability alignments is used instead

# word alignment

# Word Alignment

Given a sentence pair, which words correspond to each other?

	michael	geht	davon	aus	,	dass	er	im	haus	bleibt
michael	■									
assumes		■	■	■						
that						■				
he							■			
will										■
stay										■
in								■		
the								■		
house									■	

# Word Alignment?

	john	wohnt	hier	nicht
john	■			
does		?		?
not				■
live		■		
here			■	

Is the English word **does** aligned to the German **wohnt** (verb) or **nicht** (negation) or neither?

# Word Alignment?

	john	biss	ins	grass
john	■			
kicked		■	■	■
the		■	■	■
bucket		■	■	■

How do the idioms **kicked the bucket** and **biss ins grass** match up?  
Outside this exceptional context, **bucket** is never a good translation for **grass**



# Measuring Word Alignment Quality



- Manually align corpus with *sure* ( $S$ ) and *possible* ( $P$ ) alignment points ( $S \subseteq P$ )
- Common metric for evaluation word alignments: Alignment Error Rate (AER)

$$\text{AER}(S, P; A) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

- AER = 0: alignment  $A$  matches all sure, any possible alignment points
- However: different applications require different precision/recall trade-offs

# symmetrization

# Word Alignment with IBM Models



- IBM Models create a **many-to-one** mapping
  - words are aligned using an alignment function
  - a function may return the same value for different input (one-to-many mapping)
  - a function can not return multiple values for one input (no many-to-one mapping)
- Real word alignments have **many-to-many** mappings

# Symmetrization

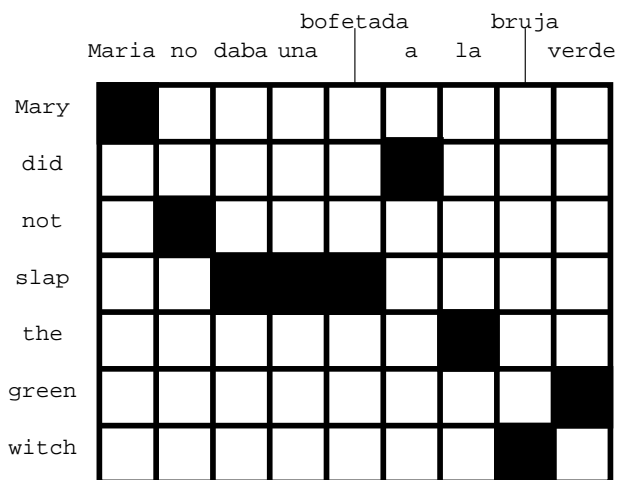
- Run IBM Model training in both directions

→ two sets of word alignment points

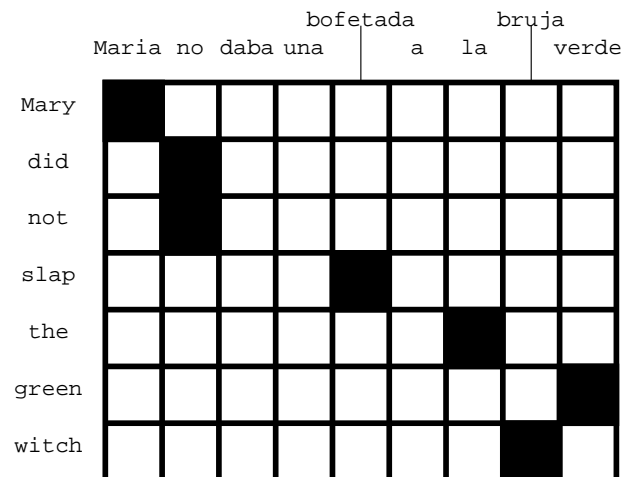
- Intersection: high precision alignment points
- Union: high recall alignment points
- Refinement methods explore the sets between intersection and union

# Example

english to spanish



spanish to english



intersection



# Growing Heuristics



**black:** intersection

**grey:** additional points in union

- Add alignment points from union based on heuristics:
  - directly/diagonally neighboring points
  - finally, add alignments that connect unaligned words in source and/or target
- Popular method: grow-diag-final-and