
Winter School

Day 2: Word-based models and the EM algorithm

MT Marathon

27 Jan 2009



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: During all the lectures, we will translate from a foreign language into English

Collect statistics

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

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

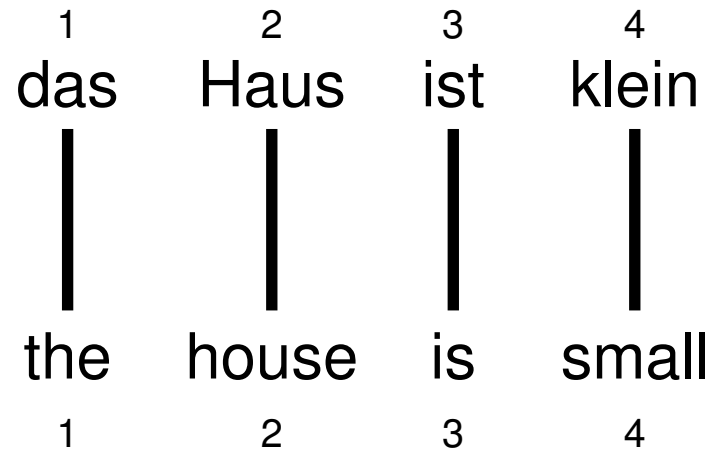
Estimate translation probabilities

- *Maximum likelihood estimation*

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

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

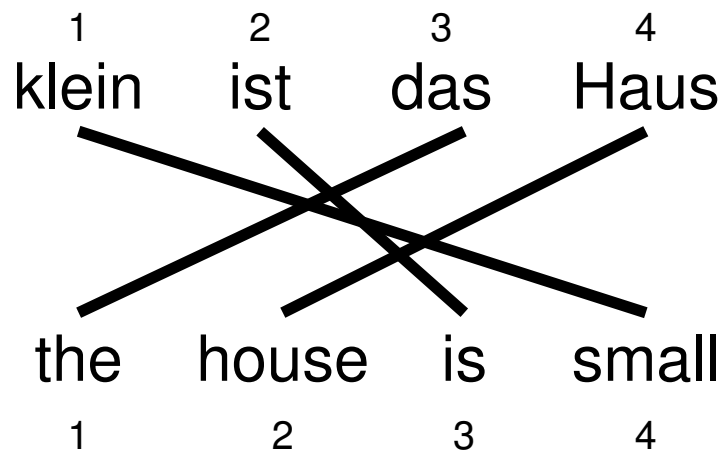
Alignment function

- 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

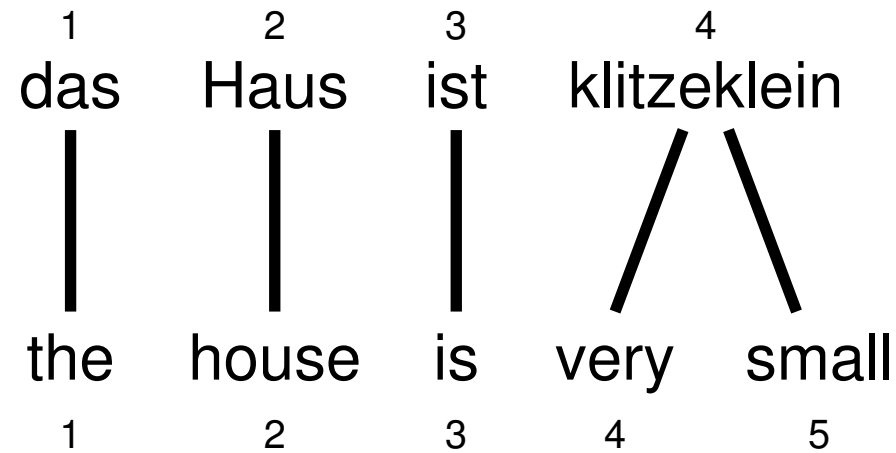
- Words may be **reordered** during translation



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

One-to-many translation

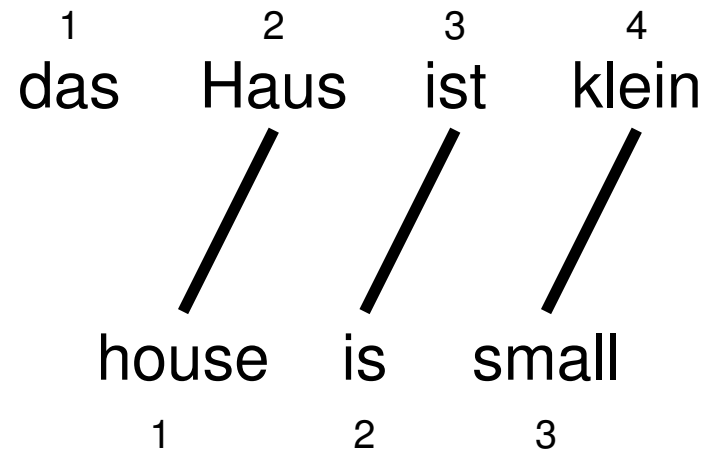
- A source word may translate into **multiple** target words



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

Dropping words

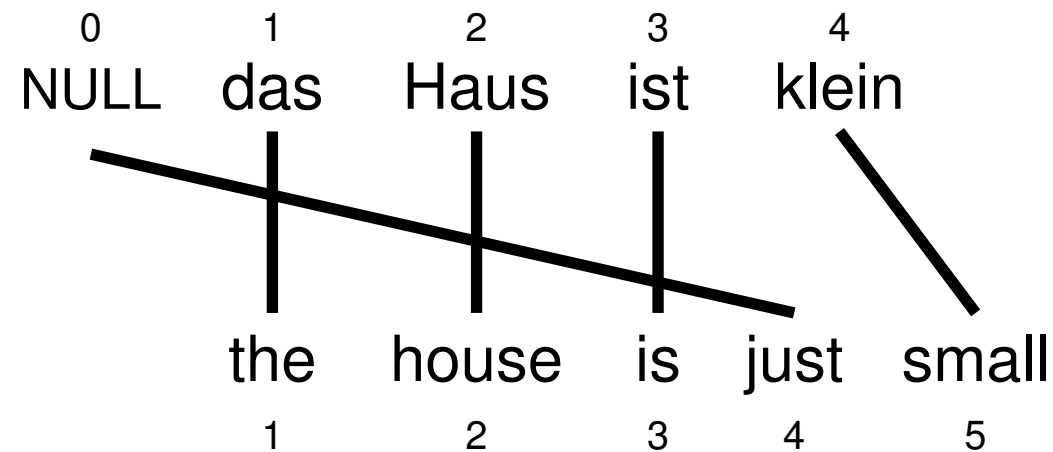
- Words may be **dropped** when translated
 - The German article *das* is dropped



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

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



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

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 ϵ is a *normalization constant*

Example

<i>das</i>		<i>Haus</i>		<i>ist</i>		<i>klein</i>	
<i>e</i>	$t(e f)$	<i>e</i>	$t(e f)$	<i>e</i>	$t(e f)$	<i>e</i>	$t(e f)$
<i>the</i>	0.7	<i>house</i>	0.8	<i>is</i>	0.8	<i>small</i>	0.4
<i>that</i>	0.15	<i>building</i>	0.16	<i>'s</i>	0.16	<i>little</i>	0.4
<i>which</i>	0.075	<i>home</i>	0.02	<i>exists</i>	0.02	<i>short</i>	0.1
<i>who</i>	0.05	<i>household</i>	0.015	<i>has</i>	0.015	<i>minor</i>	0.06
<i>this</i>	0.025	<i>shell</i>	0.005	<i>are</i>	0.005	<i>petty</i>	0.04

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



Learning lexical translation models

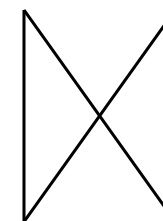
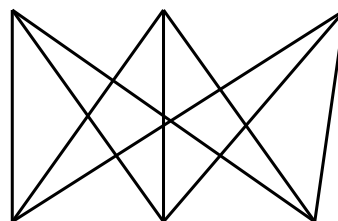
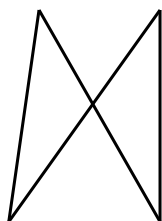
- We would like to *estimate* the lexical translation probabilities $t(e|f)$ from a parallel corpus
- ... but we do not have the alignments
- **Chicken and egg problem**
 - if we had the *alignments*,
 - we could estimate the *parameters* of our generative model
 - if we had the *parameters*,
 - we could 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
 - initialize model parameters (e.g. uniform)
 - assign probabilities to the missing data
 - estimate model parameters from completed data
 - iterate

EM algorithm

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

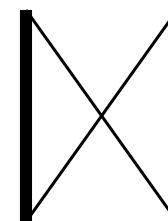
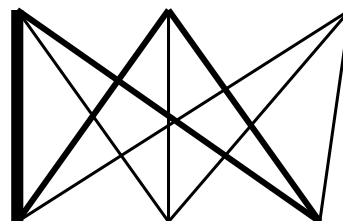
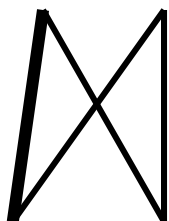


... the house ... the blue house ... the flower ...

- 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 ...

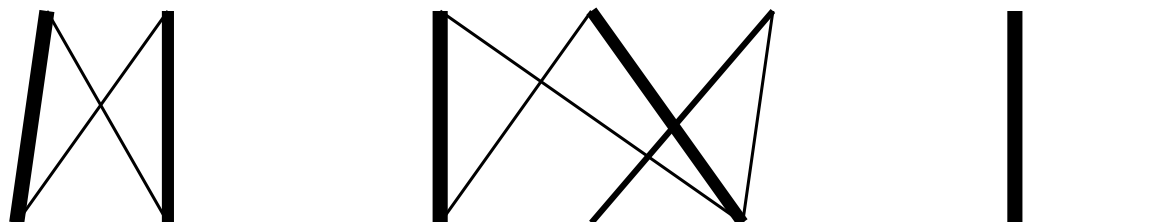


... the house ... the blue house ... the flower ...

- 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 ...

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

Diagram illustrating word alignment between French and English sentences. Vertical lines connect aligned words, and a large 'X' marks a misalignment.

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

- 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: estimate translation probabilities from weighted counts

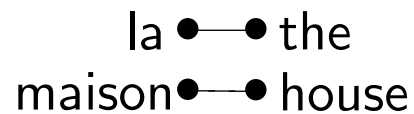
IBM Model 1 and EM

- **Probabilities**

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

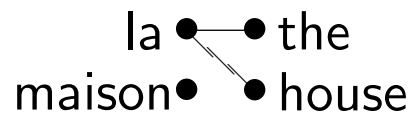
$$p(\text{the}|\text{maison}) = 0.1 \quad p(\text{house}|\text{maison}) = 0.8$$

- **Alignments**



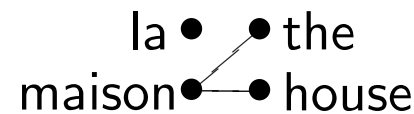
$$p(\mathbf{e}, \mathbf{a}|\mathbf{f}) = 0.56$$

$$p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = 0.824$$



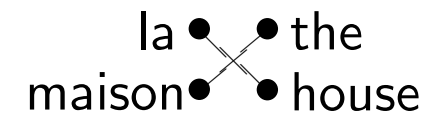
$$p(\mathbf{e}, \mathbf{a}|\mathbf{f}) = 0.035$$

$$p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = 0.052$$



$$p(\mathbf{e}, \mathbf{a}|\mathbf{f}) = 0.08$$

$$p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = 0.118$$



$$p(\mathbf{e}, \mathbf{a}|\mathbf{f}) = 0.005$$

$$p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = 0.007$$

- **Counts**

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

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

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

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

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)

IBM Model 1 and EM: Expectation Step

- 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 $l_e = l_f = 2$)

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

IBM Model 1 and EM: Expectation Step

- Combine what we have:

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

IBM Model 1 and EM: Maximization Step

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

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

- Using the expression on the previous slide, and noting that only alignments which link e and f are relevant, we obtain:

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

IBM Model 1 and EM: Maximization Step

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

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

IBM Model 1 and EM: Pseudocode

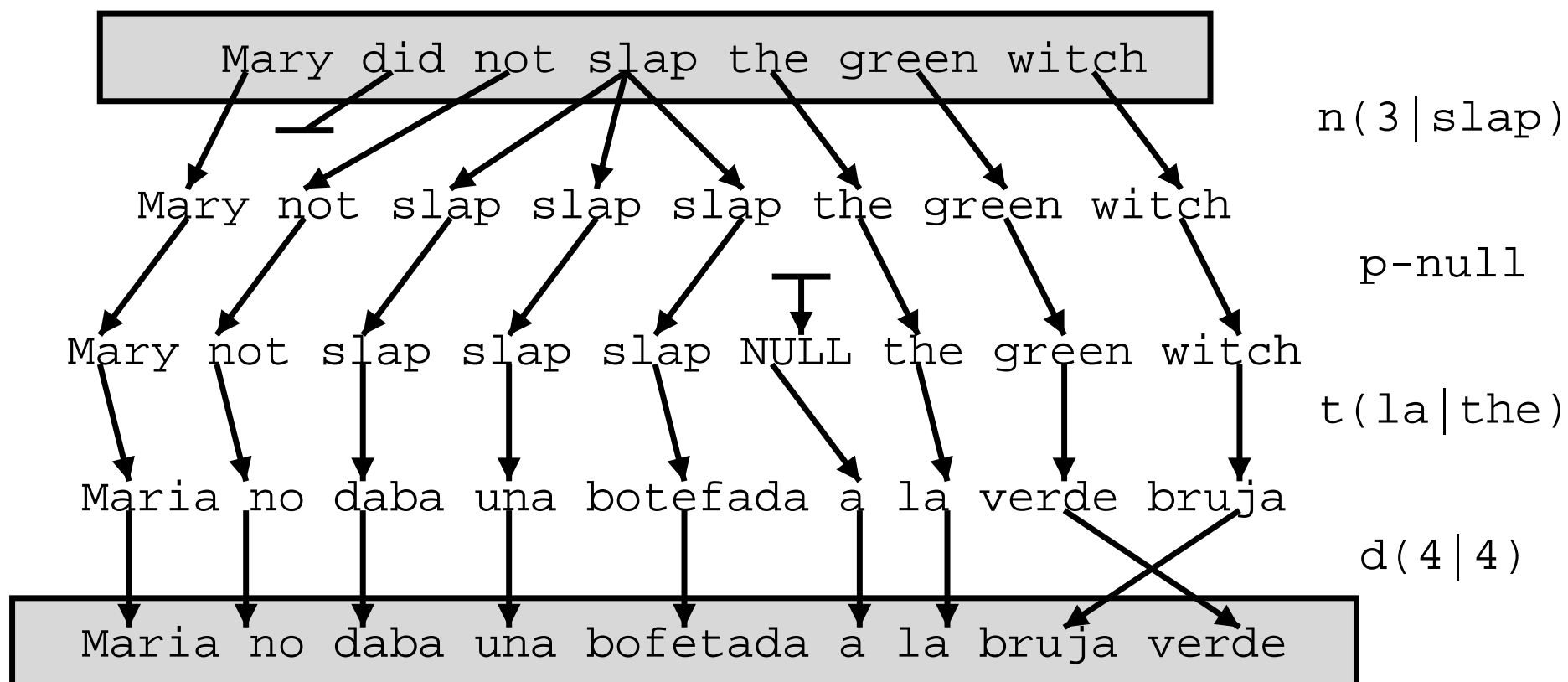
```
initialize  $t(e|f)$  uniformly
do until convergence
  set  $\text{count}(e|f)$  to 0 for all  $e, f$ 
  set  $\text{total}(f)$  to 0 for all  $f$ 
  for all sentence pairs  $(e\_s, f\_s)$ 
    for all words  $e$  in  $e\_s$ 
       $\text{total\_s}(e) = 0$ 
      for all words  $f$  in  $f\_s$ 
         $\text{total\_s}(e) += t(e|f)$ 
    for all words  $e$  in  $e\_s$ 
      for all words  $f$  in  $f\_s$ 
         $\text{count}(e|f) += t(e|f) / \text{total\_s}(e)$ 
         $\text{total}(f) += t(e|f) / \text{total\_s}(e)$ 
  for all  $f$ 
    for all  $e$ 
       $t(e|f) = \text{count}(e|f) / \text{total}(f)$ 
```

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

IBM Model 4



Word alignment

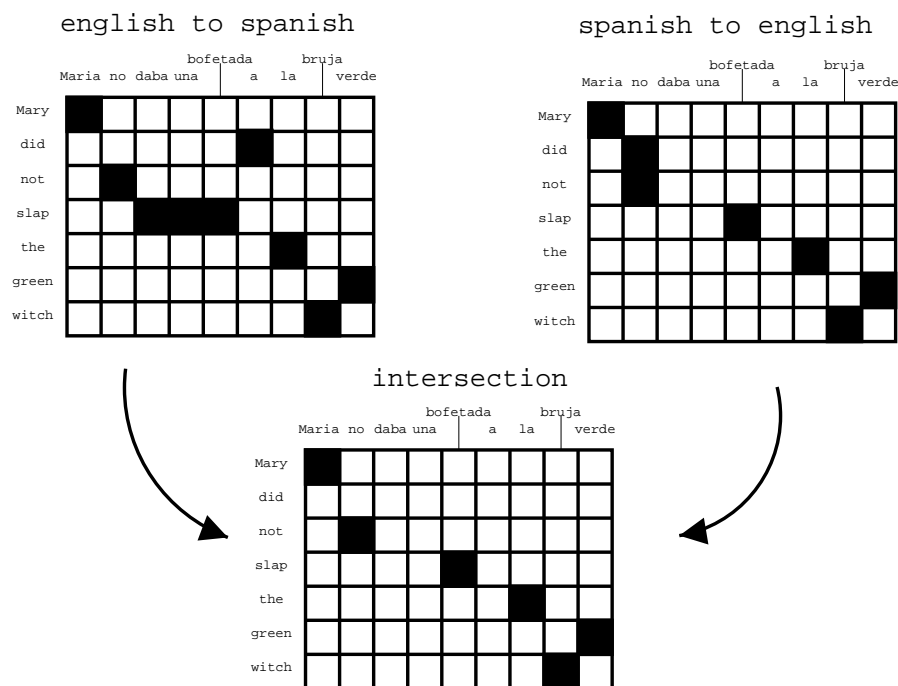
- IBM Models are nowadays mainly used for word alignment
- Other word alignment models proposed e.g. HMM
- Shared task at NAACL 2003 and ACL 2005 workshops

	Mar	no	daba	una	bofetada	a	la	bruja	verde
Mary	■								
did		■							
not									
slap			■	■	■				
the						■	■		
green									■
witch								■	

Word alignment with IBM models

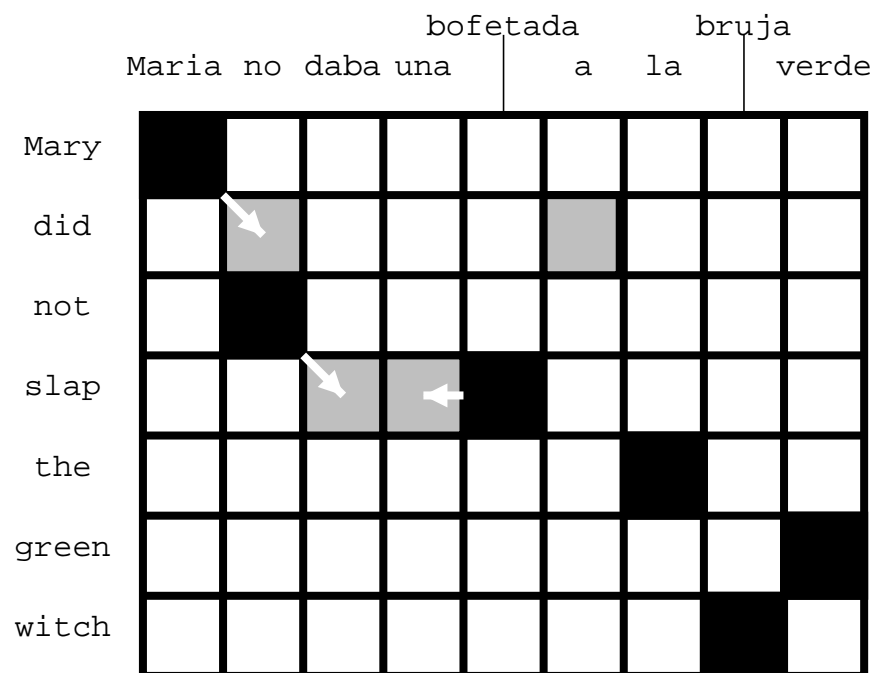
- IBM Models create a *many-to-one* mapping
 - words are aligned using an **alignment function**
 - a function may return the same value for different input (one-to-many mapping)
 - a function can not return multiple values for one input (*no many-to-one* mapping)
- But we need *many-to-many* mappings

Symmetrizing word alignments



- *Intersection* of GIZA++ bidirectional alignments

Symmetrizing word alignments



- *Grow* additional alignment points [Och and Ney, CompLing2003]



Growing heuristic

```
GROW-DIAG-FINAL-AND(e2f,f2e):
  neighboring = ((-1,0),(0,-1),(1,0),(0,1),(-1,-1),(-1,1),(1,-1),(1,1))
  alignment = intersect(e2f,f2e);
  GROW-DIAG(); FINAL-AND(e2f); FINAL-AND(f2e);
```

```
GROW-DIAG():
  iterate until no new points added
  for english word e = 0 ... en
    for foreign word f = 0 ... fn
      if ( e aligned with f )
        for each neighboring point ( e-new, f-new ):
          if ( ( e-new not aligned or f-new not aligned ) and
              ( e-new, f-new ) in union( e2f, f2e ) )
            add alignment point ( e-new, f-new )
```

```
FINAL-AND(a):
  for english word e-new = 0 ... en
    for foreign word f-new = 0 ... fn
      if ( ( e-new not aligned and f-new not aligned ) and
          ( e-new, f-new ) in alignment a )
        add alignment point ( e-new, f-new )
```

More Recent Work

- Symmetrization during training
 - symmetrize after each iteration of IBM Models
 - integrate symmetrization into models
 - e.g. Liang, Taskar and Klein, NAACL 2006
- Discriminative training methods
 - supervised learning based on labeled data
 - semi-supervised learning with limited labeled data
 - e.g. Blunsom and Cohn, ACL 2006
- Better generative models
 - e.g. Fraser and Marcu, EMNLP 2007