
Empirical Methods in Natural Language Processing

Lecture 15

Machine translation (II): Word-based models and the EM algorithm

Philipp Koehn

26 February 2007



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

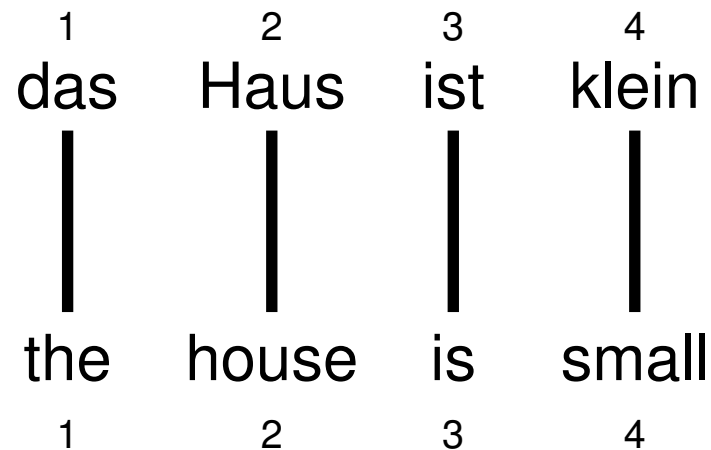
Estimate translation probabilities

- *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

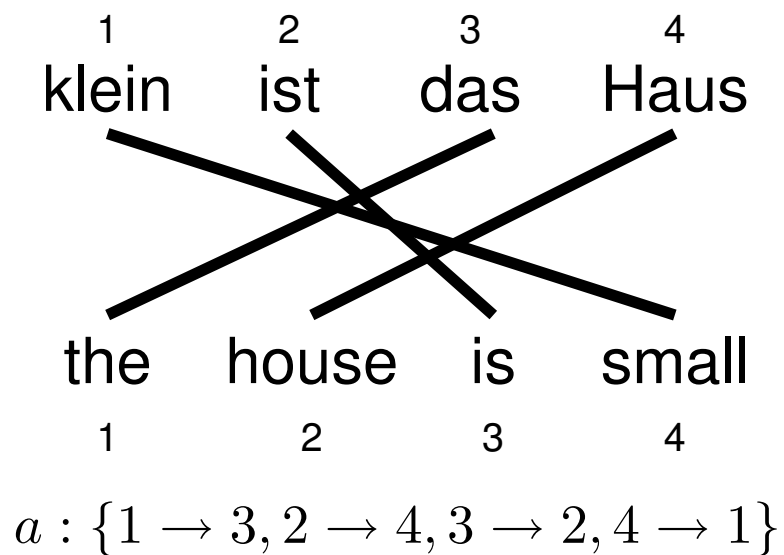
Alignment function

- 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

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

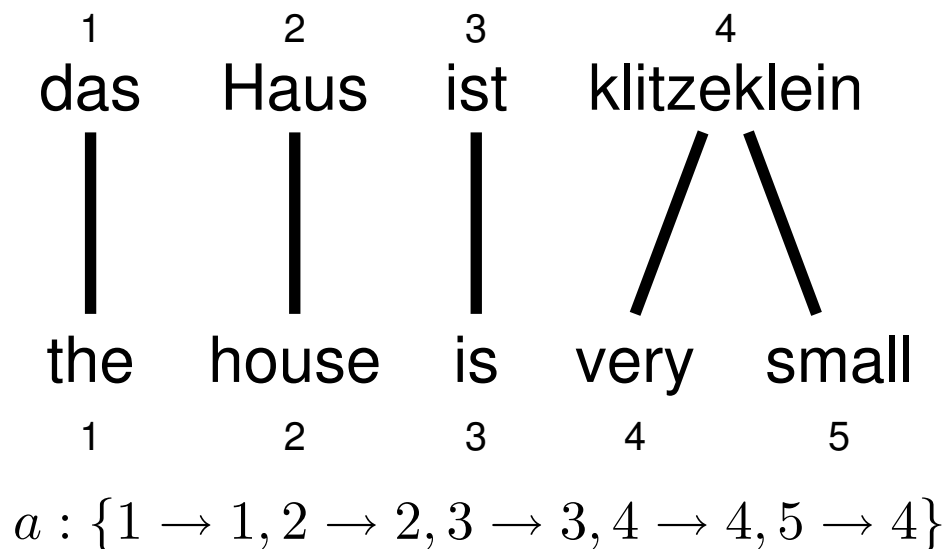
Reordering

- Words may be **reordered** during translation



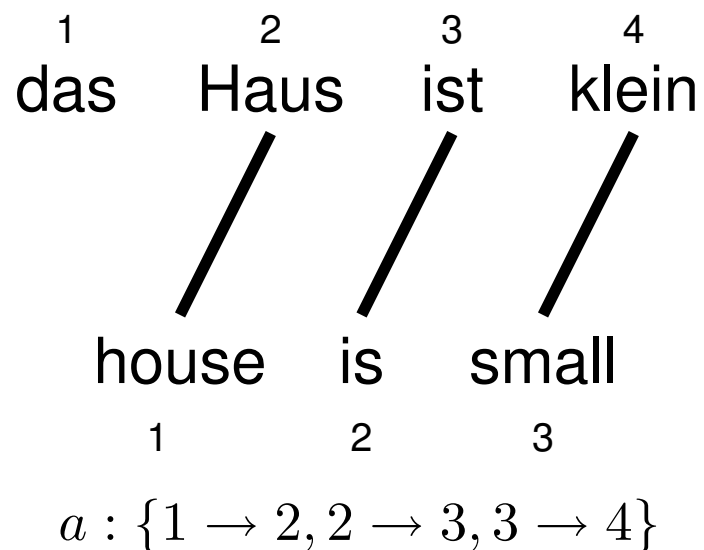
One-to-many translation

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



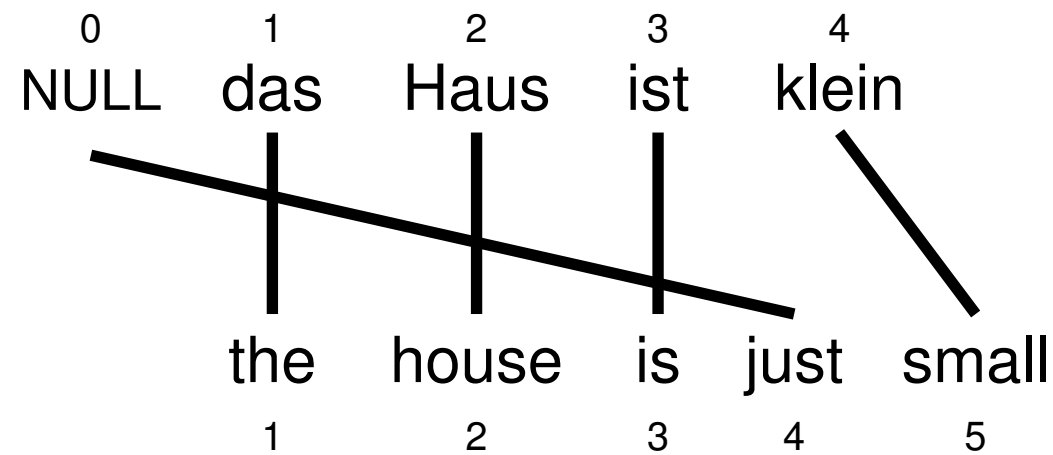
Dropping words

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



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

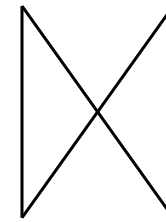
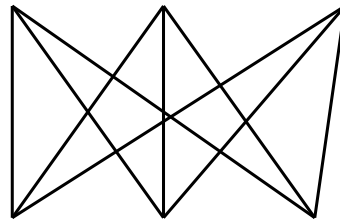
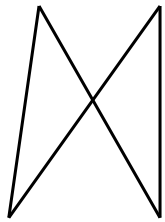
- We would like to *estimate* the lexical translation probabilities $t(e|f)$ from a parallel corpus
- ... but we do not have the alignments
- **Chicken and egg problem**
 - if we had the *alignments*,
 - we could estimate the *parameters* of our generative model
 - if we had the *parameters*,
 - we could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

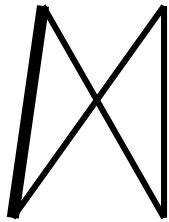


... the house ... the blue house ... the flower ...

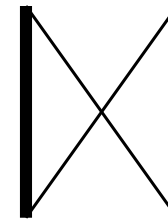
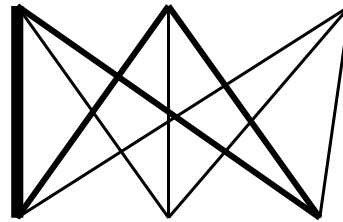
- Initial step: all alignments equally likely
- Model learns that, e.g., *la* is often aligned with *the*

EM algorithm

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

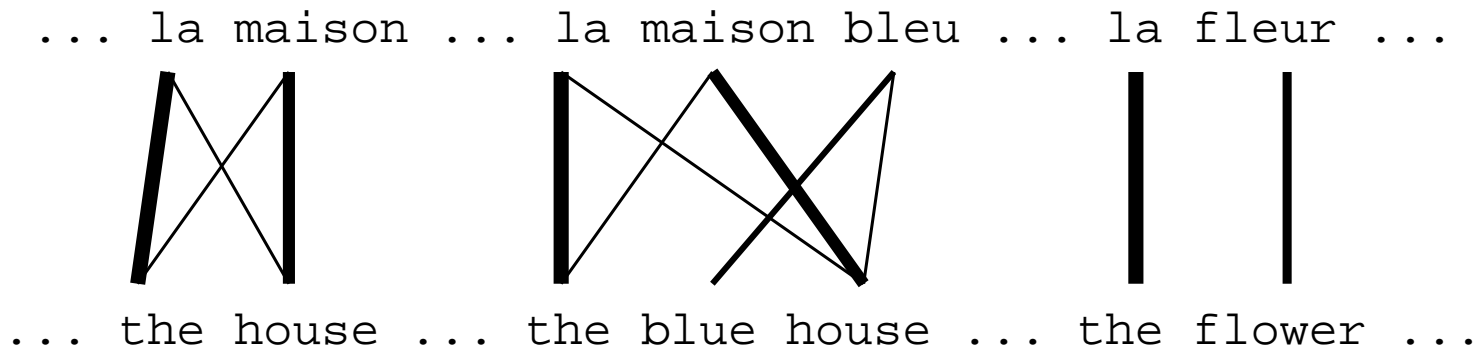


... the house ... the blue house ... the flower ...



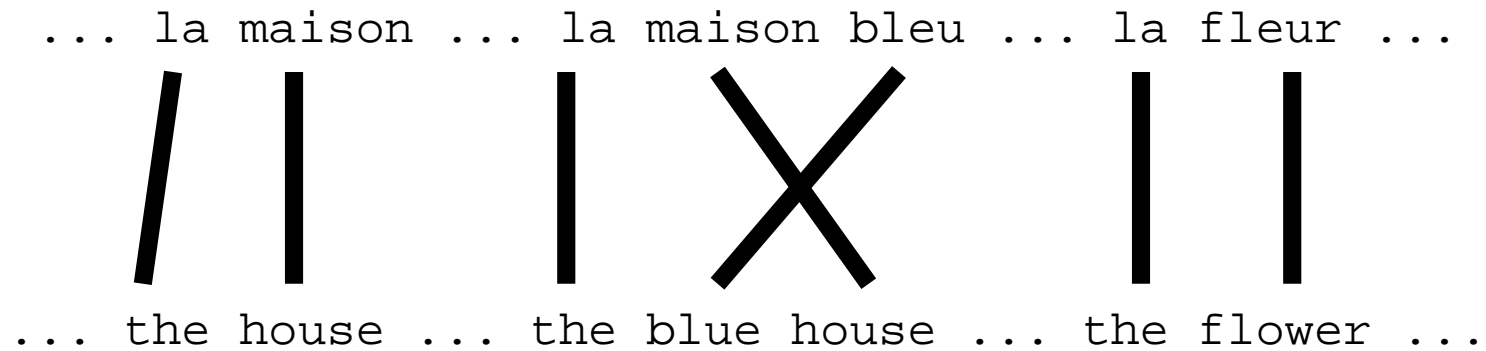
- After one iteration
- Alignments, e.g., between *la* and *the* are more likely

EM algorithm



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

EM algorithm



- 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

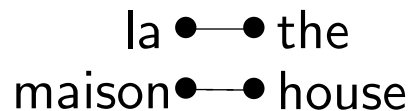
- We need to be able to compute:
 - Expectation-Step: probability of alignments
 - Maximization-Step: count collection

IBM Model 1 and EM

- Probabilities**

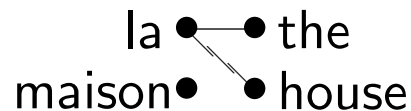
$$\begin{aligned}
 p(\text{the}|\text{la}) &= 0.7 & p(\text{house}|\text{la}) &= 0.05 \\
 p(\text{the}|\text{maison}) &= 0.1 & p(\text{house}|\text{maison}) &= 0.8
 \end{aligned}$$

- Alignments**



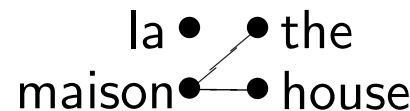
$$p(\mathbf{e}, \mathbf{a}|\mathbf{f}) = 0.56$$

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



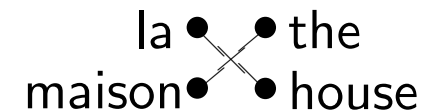
$$p(\mathbf{e}, \mathbf{a}|\mathbf{f}) = 0.035$$

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



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

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



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

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

- Counts**

$$\begin{aligned}
 c(\text{the}|\text{la}) &= 0.824 + 0.052 & c(\text{house}|\text{la}) &= 0.052 + 0.007 \\
 c(\text{the}|\text{maison}) &= 0.118 + 0.007 & c(\text{house}|\text{maison}) &= 0.824 + 0.118
 \end{aligned}$$

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

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

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

IBM Model 1 and EM: Expectation Step

- 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

- 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:

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

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
  set count( $e|f$ ) to 0 for all  $e, f$ 
  set total( $f$ ) to 0 for all  $f$ 
  for all sentence pairs ( $e_s, f_s$ )
    for all words  $e$  in  $e_s$ 
      total_s = 0
      for all words  $f$  in  $f_s$ 
        total_s +=  $t(e|f)$ 
      for all words  $e$  in  $e_s$ 
        for all words  $f$  in  $f_s$ 
          count( $e|f$ ) +=  $t(e|f) / \text{total}_s$ 
          total( $f$ ) +=  $t(e|f) / \text{total}_s$ 
    for all  $f$  in domain( total(.) )
      for all  $e$  in domain( count(.|f) )
         $t(e|f) = \text{count}(e|f) / \text{total}(f)$ 
until convergence
```

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

