Chapter 4

Word-based models

Statistical Machine Translation

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)

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

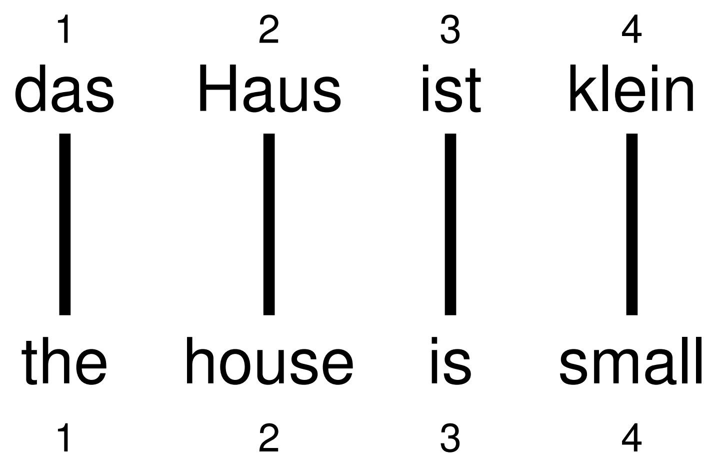
Estimate Translation Probabilities

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

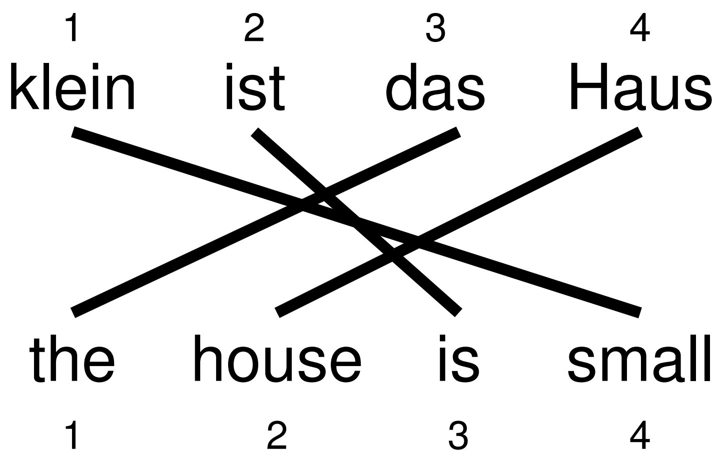
- Mapping an English target word at position i to a German source word at position j with a function $a : i \to j$

- Example

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

Reordering

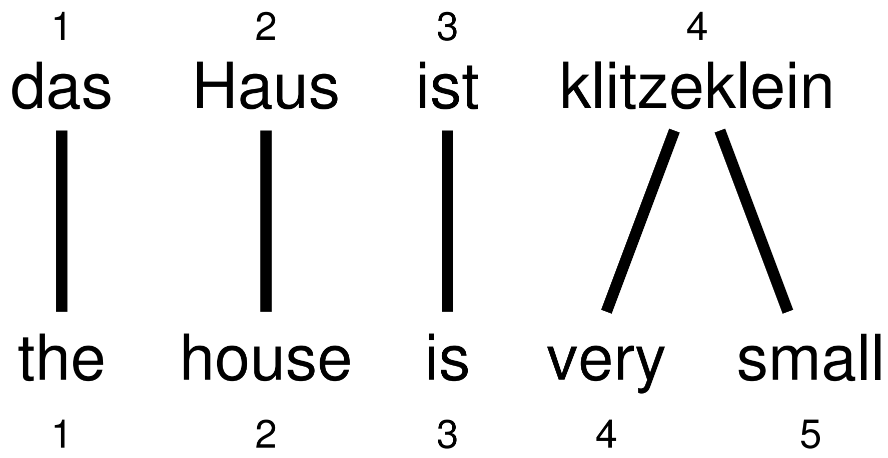
Words may be reordered during translation



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

One-to-Many Translation

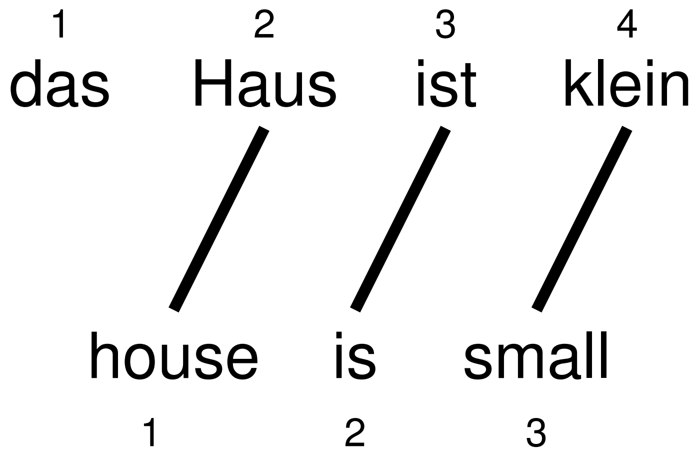
A source word may translate into multiple target words



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

Dropping Words

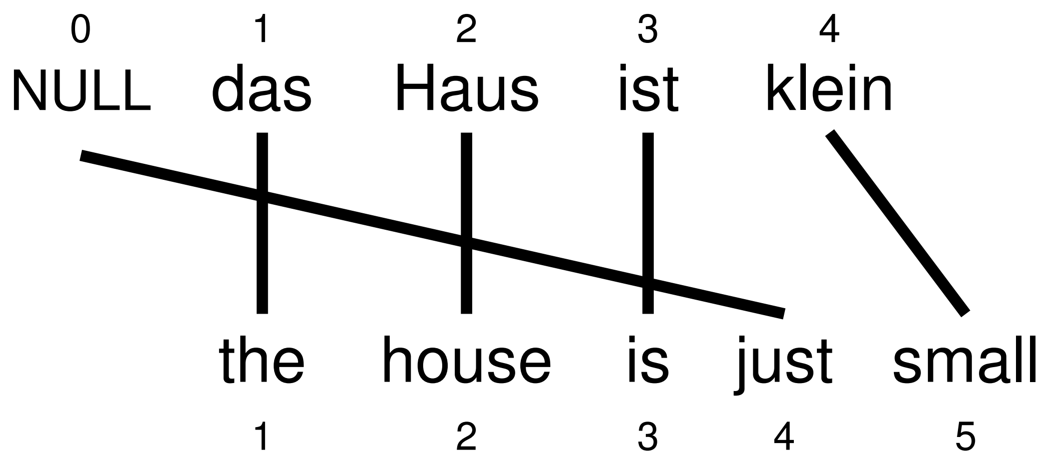
Words may be dropped when translated
(German article das is dropped)



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

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 $\mathbf{f} = (f_1, ..., f_{l_f})$ 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 $a : j \to i$

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

  - parameter ϵ is a normalization constant

Example

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

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

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

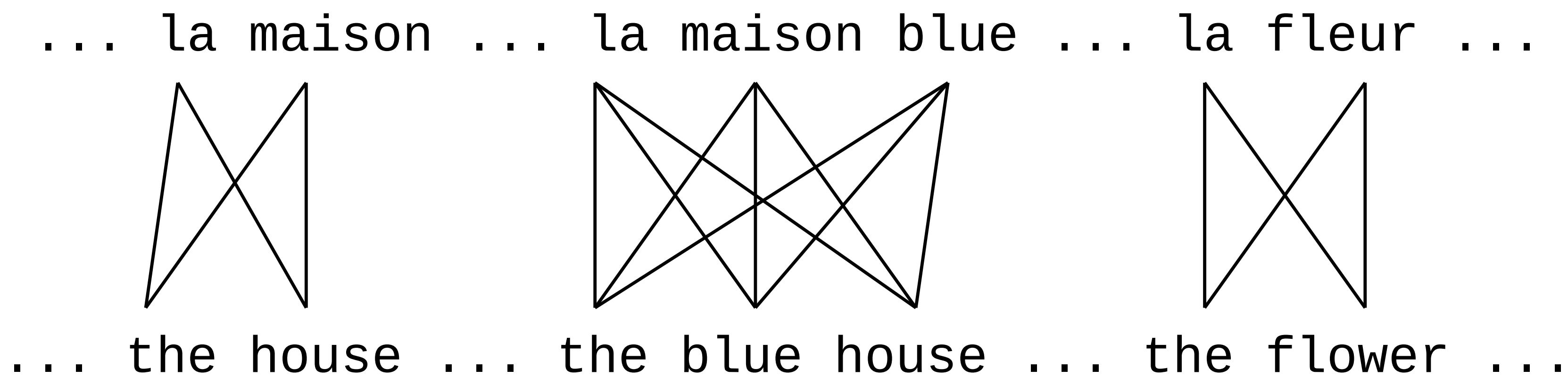
- Chicken and egg problem
  - if we had the *alignments*,
    $\rightarrow$ we could estimate the *parameters* of our generative model
  - if we had the *parameters*,
    $\rightarrow$ we could estimate the *alignments*

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*

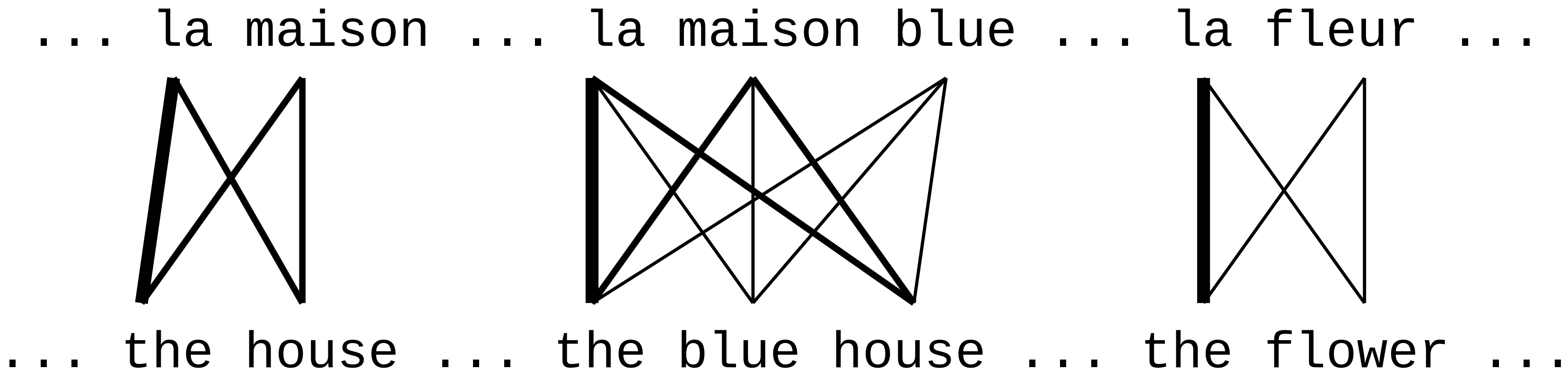
- Expectation Maximization (EM) in a nutshell
  1. initialize model parameters (e.g. uniform)
  2. assign probabilities to the missing data
  3. estimate model parameters from completed data
  4. iterate steps 2–3 until convergence

EM Algorithm



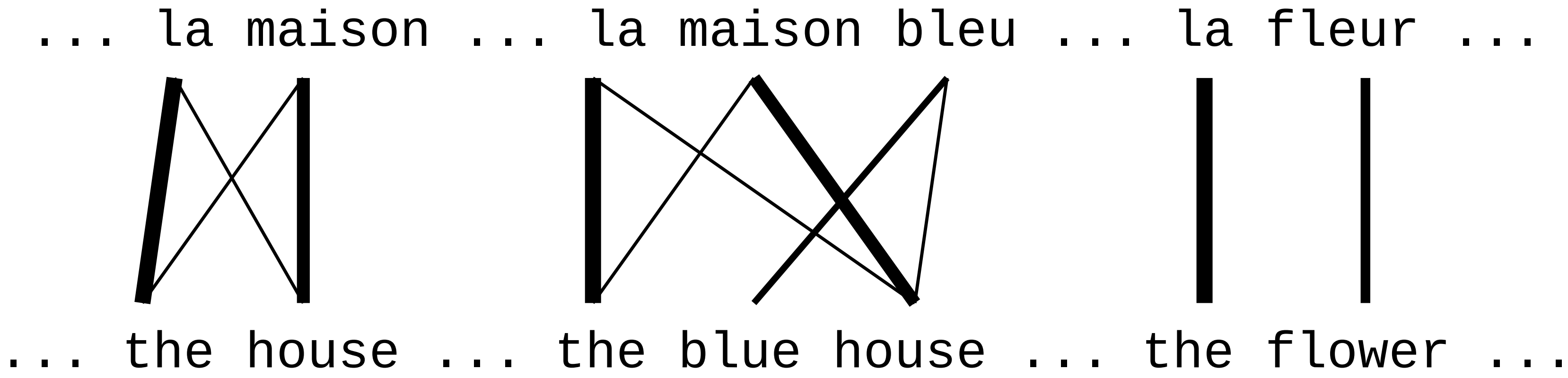
- Initial step: all alignments equally likely
- Model learns that, e.g., la is often aligned with the

EM Algorithm



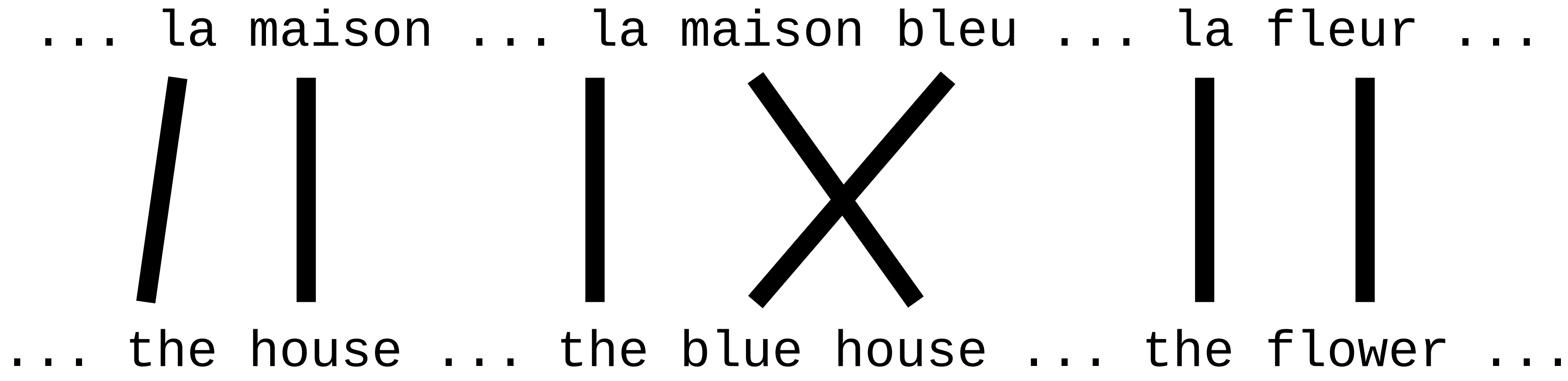
- After one iteration
- Alignments, e.g., between la and the are more likely

EM Algorithm



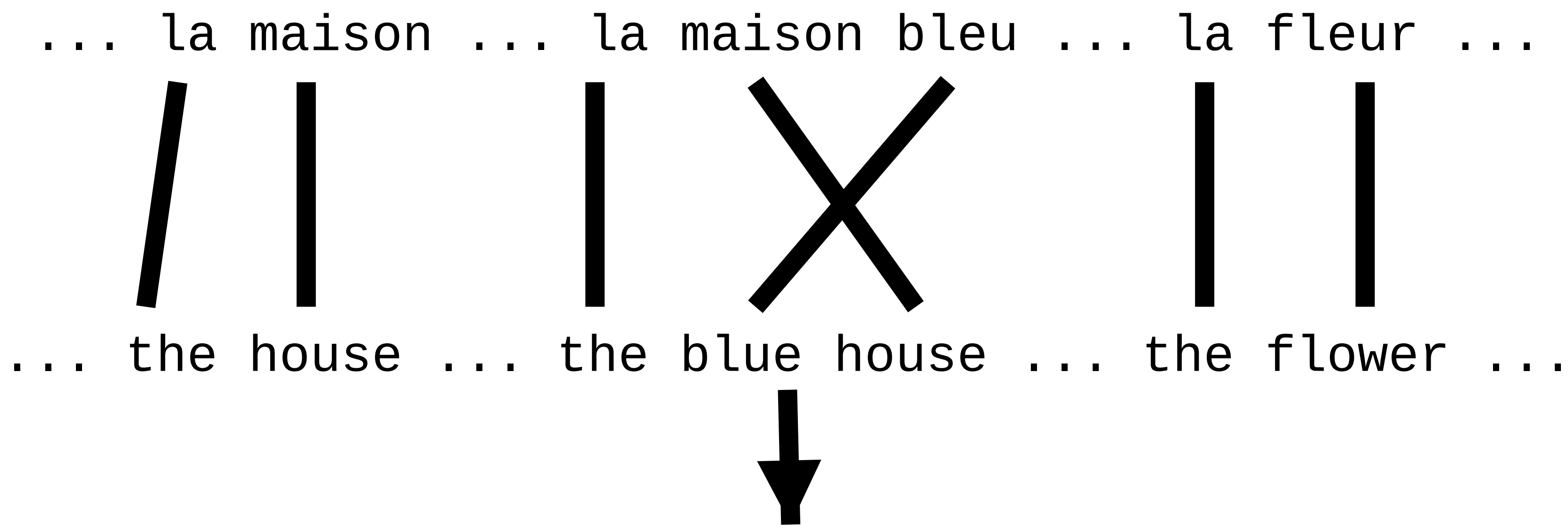
- After another iteration
- It becomes apparent that alignments, e.g., between fleur and flower are more likely (pigeon hole principle)

EM Algorithm



- Convergence
- Inherent hidden structure revealed by EM

EM Algorithm




```
p(la|the) = 0.453
p(le|the) = 0.334
p(maison|house) = 0.876
p(bleu|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

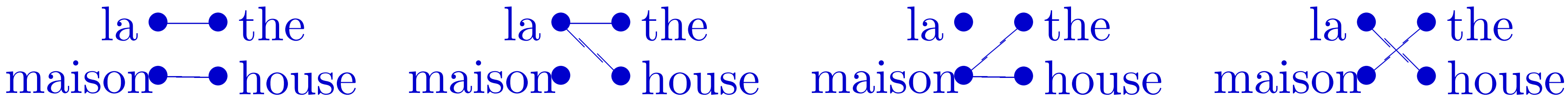
- We need to be able to compute:
  - Expectation-Step: probability of alignments
  - Maximization-Step: count collection

IBM Model 1 and EM

- **Probabilities**

$$p(\text{the}|\text{la}) = 0.7 \qquad p(\text{house}|\text{la}) = 0.05$$
$$p(\text{the}|\text{maison}) = 0.1 \qquad p(\text{house}|\text{maison}) = 0.8$$

- **Alignments**



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

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

- **Counts**

$$c(\text{the}|\text{la}) = 0.824 + 0.052 \qquad c(\text{house}|\text{la}) = 0.052 + 0.007$$
$$c(\text{the}|\text{maison}) = 0.118 + 0.007 \qquad c(\text{house}|\text{maison}) = 0.824 + 0.118$$

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

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

- Note the trick in the last line
  - removes the need for an exponential number of products

$\rightarrow$ 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:

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

IBM Model 1 and EM: Maximization Step

- Now we have to collect counts

- Evidence from a sentence pair **e,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 simplication as before:

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

IBM Model 1 and EM: Maximization Step

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

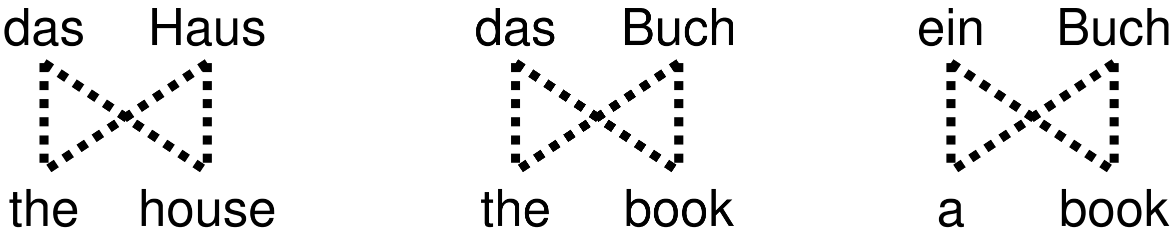
$$t(e|f;\mathbf{e},\mathbf{f}) = \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e|f;\mathbf{e},\mathbf{f}))}{\sum_e \sum_{(\mathbf{e},\mathbf{f})} c(e|f;\mathbf{e},\mathbf{f}))}$$

IBM Model 1 and EM: Pseudocode

Input: set of sentence pairs $(\mathbf{e}, \mathbf{f})$
Output: translation prob. $t(e|f)$

initialize $t(e|f)$ uniformly
while not converged **do**
    // initialize
    count$(e|f) = 0$ **for all** e, f
    total$(f) = 0$ **for all** f
    for all sentence pairs (**e**,**f**) **do**
        // compute normalization
        for all words e in **e do**
            s-total$(e) = 0$
            for all words f in **f do**
                s-total(e) $+= t(e|f)$
            end for
        end for
        // collect counts
        for all words e in **e do**
            for all words f in **f do**
                count$(e|f)$ $+= \frac{t(e|f)}{\text{s-total}(e)}$
                total(f) $+= \frac{t(e|f)}{\text{s-total}(e)}$
            end for
        end for
    end for
    // estimate probabilities
    for all foreign words f **do**
        for all English words e **do**
            $t(e|f) = \frac{\text{count}(e|f)}{\text{total}(f)}$
        end for
    end for
end while

Convergence



e	f	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 (ϵ=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

Ensuring Fluent Output

- Our translation model cannot decide between small and little
- Sometime one is preferred over the other:
  - small step: 2,070,000 occurrences in the Google index
  - little step: 257,000 occurrences in the Google index
- Language model
  - estimate how likely a string is English
  - based on n-gram statistics

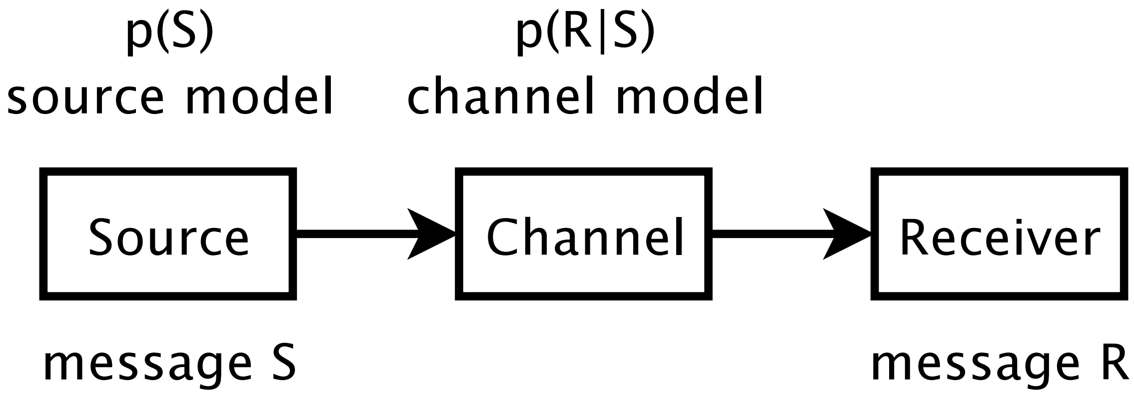
$$\begin{aligned} p(\mathbf{e}) &= p(e_1, e_2, ..., e_n) \\ &= p(e_1)p(e_2|e_1)...p(e_n|e_1, e_2, ..., e_{n-1}) \\ &\simeq p(e_1)p(e_2|e_1)...p(e_n|e_{n-2}, e_{n-1}) \end{aligned}$$

Noisy Channel Model

- We would like to integrate a language model
- Bayes rule

$$\text{argmax}_{\mathbf{e}}\, p(\mathbf{e}|\mathbf{f}) = \text{argmax}_{\mathbf{e}} \frac{p(\mathbf{f}|\mathbf{e})\, p(\mathbf{e})}{p(\mathbf{f})}$$
$$= \text{argmax}_{\mathbf{e}}\, p(\mathbf{f}|\mathbf{e})\, p(\mathbf{e})$$

Noisy Channel Model



- Applying Bayes rule also called noisy channel model
  - we observe a distorted message R (here: a foreign string **f**)
  - we have a model on how the message is distorted (here: translation model)
  - we have a model on what messages are probably (here: language model)
  - we want to recover the original message S (here: an English string **e**)

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

- Compuationally biggest change in Model 3
  - trick to simplify estimation does not work anymore
  - $\rightarrow$ exhaustive count collection becomes computationally too expensive
  - sampling over high probability alignments is used instead

Reminder: IBM Model 1

- Generative model: break up translation process into smaller steps
  - IBM Model 1 only uses lexical translation

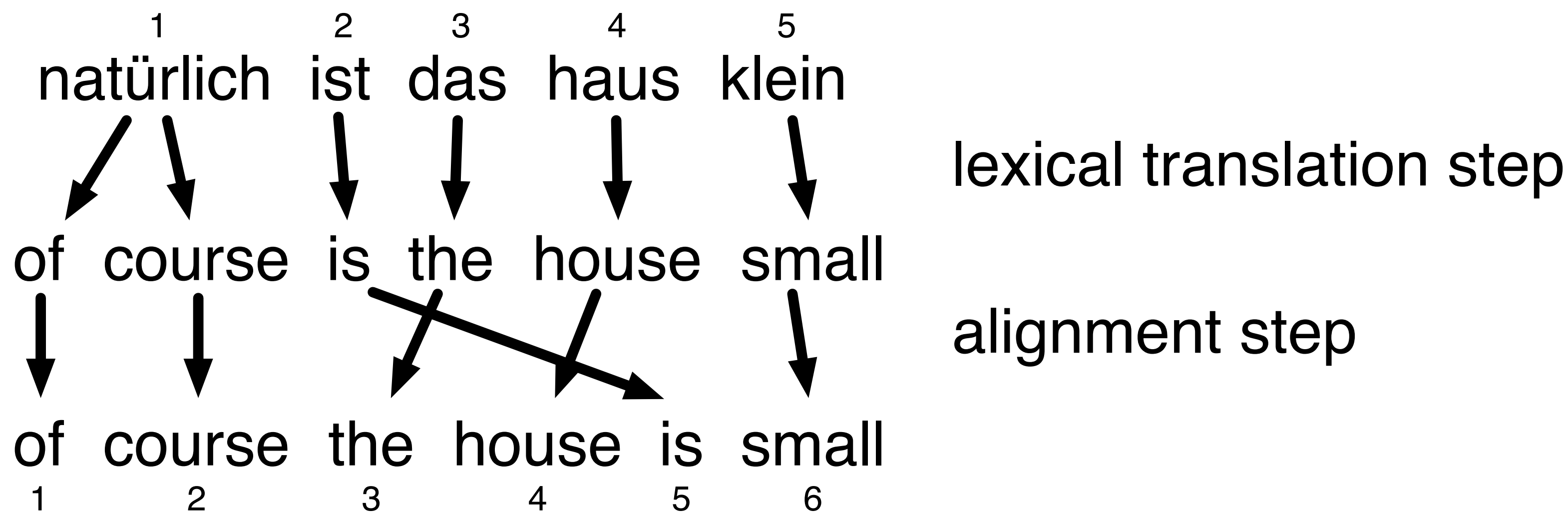
- Translation probability
  - for a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$ of length l_f
  - to an English sentence $\mathbf{e} = (e_1, ..., e_{l_e})$ of length l_e
  - with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \to i$

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

  - parameter ϵ is a normalization constant

IBM Model 2

Adding a model of alignment



IBM Model 2

- Modeling alignment with an alignment probability distribution

- Translating foreign word at position i to English word 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

Interlude: HMM Model

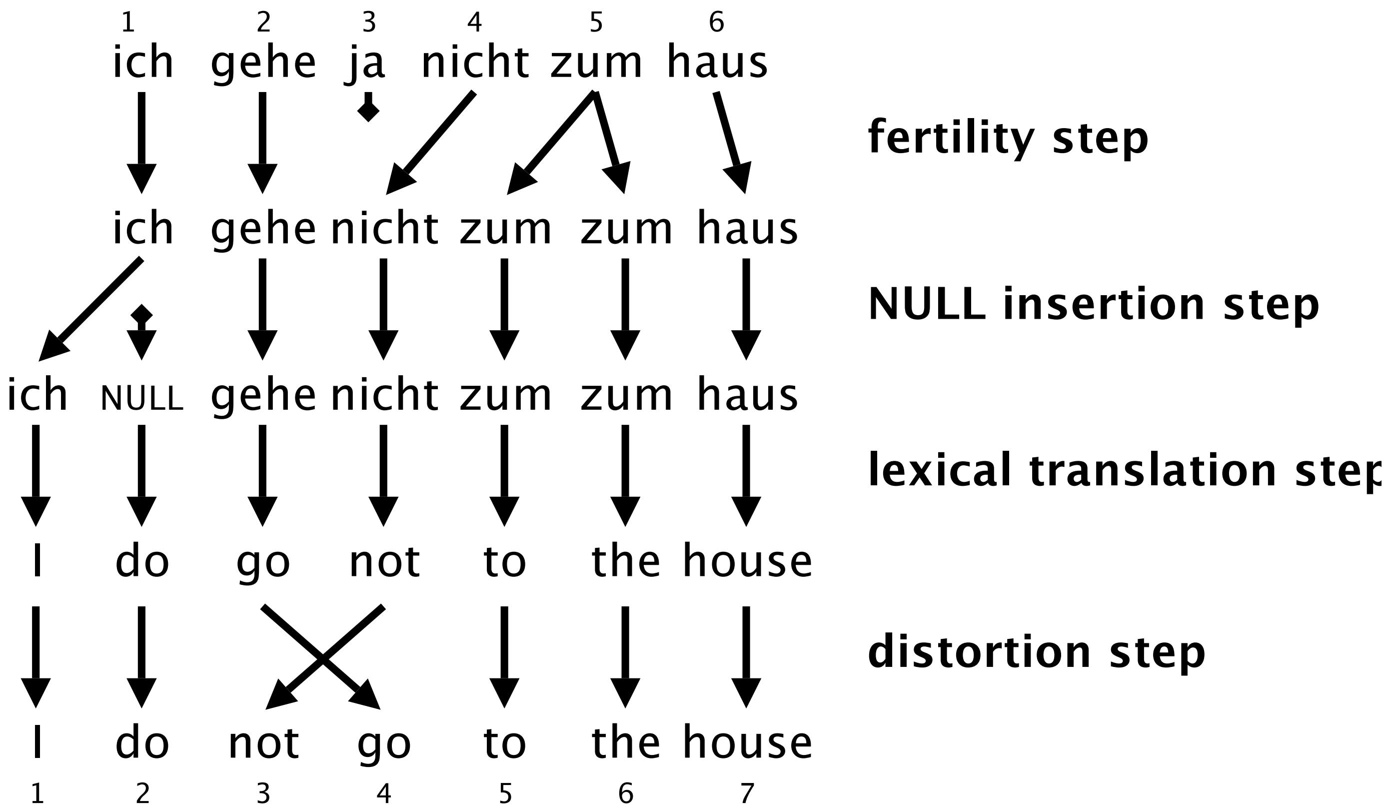
- Words do not move independently of each other
  - they often move in groups
  - $\rightarrow$ condition word movements on previous word
- HMM alignment model:

$$p(a(j)|a(j-1), l_e)$$

- EM algorithm application harder, requires dynamic programming
- IBM Model 4 is similar, also conditions on word classes

IBM Model 3

Adding a model of fertilty



IBM Model 3: Fertility

- Fertility: number of English words generated by a foreign word
- Modelled by distribution $n(\phi|f)$
- Example:

$$
\begin{aligned}
n(1|\text{haus}) &\simeq 1 \\
n(2|\text{zum}) &\simeq 1 \\
n(0|\text{ja}) &\simeq 1
\end{aligned}
$$

Sampling the Alignment Space

- Training IBM Model 3 with the EM algorithm
  - The trick that reduces exponential complexity does not work anymore
  - $\rightarrow$ Not possible to exhaustively consider all alignments

- Finding the most probable alignment by hillclimbing
  - start with initial alignment
  - change alignments for individual words
  - keep change if it has higher probability
  - continue until convergence

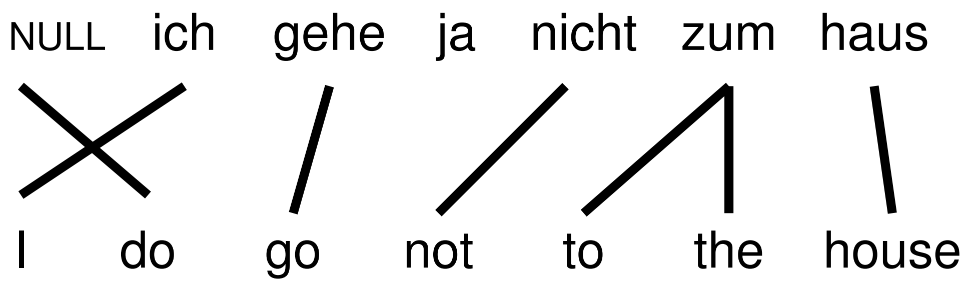
- Sampling: collecting variations to collect statistics
  - all alignments found during hillclimbing
  - neighboring alignments that differ by a move or a swap

IBM Model 4

- Better reordering model
- Reordering in IBM Model 2 and 3
  - recall: $d(j||_i, l_e, lf)$
  - for large sentences (large l_f and l_e), sparse and unreliable statistics
  - phrases tend to move together
- Relative reordering model: relative to previously translated words (cepts)

IBM Model 4: Cepts

Foreign words with non-zero fertility forms cepts
(here 5 cepts)



cept π_i	π_1	π_2	π_3	π_4	π_5
foreign position $[i]$	1	2	4	5	6
foreign word $f_{[i]}$	ich	gehe	nicht	zum	haus
English words $\{e_j\}$	I	go	not	to,the	house
English positions $\{j\}$	1	4	3	5,6	7
center of cept $\odot_i$	1	4	3	6	7

IBM Model 4: Relative Distortion

j	1	2	3	4	5	6	7
e_j	I	do	not	go	to	the	house
in cept $\pi_{i,k}$	$\pi_{1,0}$	$\pi_{0,0}$	$\pi_{3,0}$	$\pi_{2,0}$	$\pi_{4,0}$	$\pi_{4,1}$	$\pi_{5,0}$
$\odot_{i-1}$	0	-	4	1	3	-	6
$j - \odot_{i-1}$	$+1$	-	-1	$+3$	$+2$	-	$+1$
distortion	$d_1(+1)$	1	$d_1(-1)$	$d_1(+3)$	$d_1(+2)$	$d_{>1}(+1)$	$d_1(+1)$

- Center $\odot_i$ of a cept π_i is ceiling(avg(j))
- Three cases:
  - uniform for NULL generated words
  - first word of a cept: d_1
  - next words of a cept: $d_{>1}$

Word Classes

- Some words may trigger reordering $\rightarrow$ condition reordering on words

for initial word in cept: $d_1(j - \odot_{[i-1]} | f_{[i-1]}, e_j)$

for additional words: $d_{>1}(j - \Pi_{i,k-1} | e_j)$

- Sparse data concerns $\rightarrow$ cluster words into classes

for initial word in cept: $d_1(j - \odot_{[i-1]} | \mathcal{A}(f_{[i-1]}), \mathcal{B}(e_j))$

for additional words: $d_{>1}(j - \Pi_{i,k-1} | \mathcal{B}(e_j))$

IBM Model 5

- IBM Models 1–4 are *deficient*
  - some impossible translations have positive probability
  - multiple output words may be placed in the same position

  $\rightarrow$ probability mass is wasted

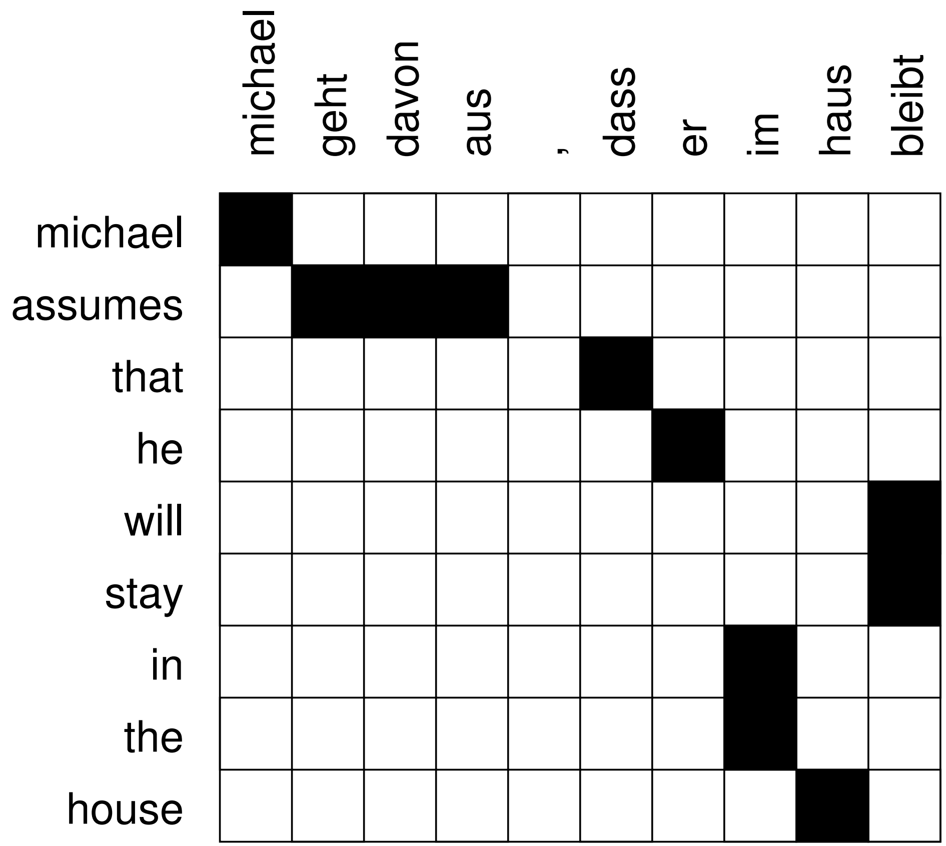
- IBM Model 5 fixes deficiency by keeping track of vacancies (available positions)

Conclusion

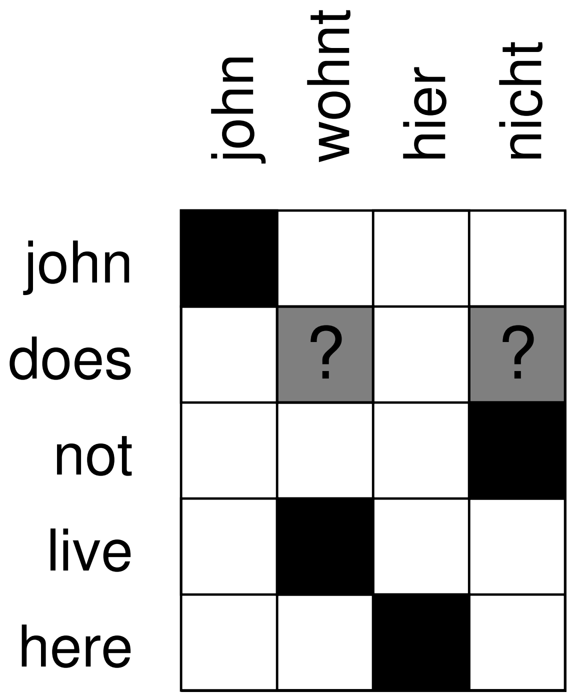
- IBM Models were the pioneering models in statistical machine translation
- Introduced important concepts
  - generative model
  - EM training
  - reordering models
- Only used for niche applications as translation model
- ... but still in common use for word alignment (e.g., GIZA++ toolkit)

Word Alignment

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

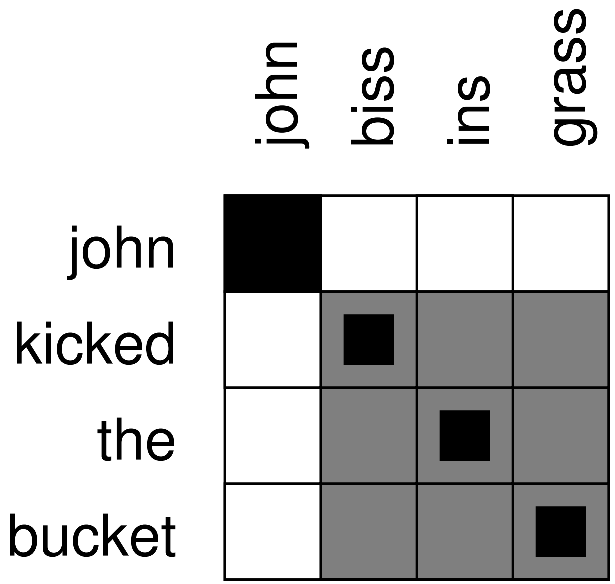


Word Alignment?



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

Word Alignment?



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)

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

- AER $= 0$: alignment A matches all sure, any possible alignment points

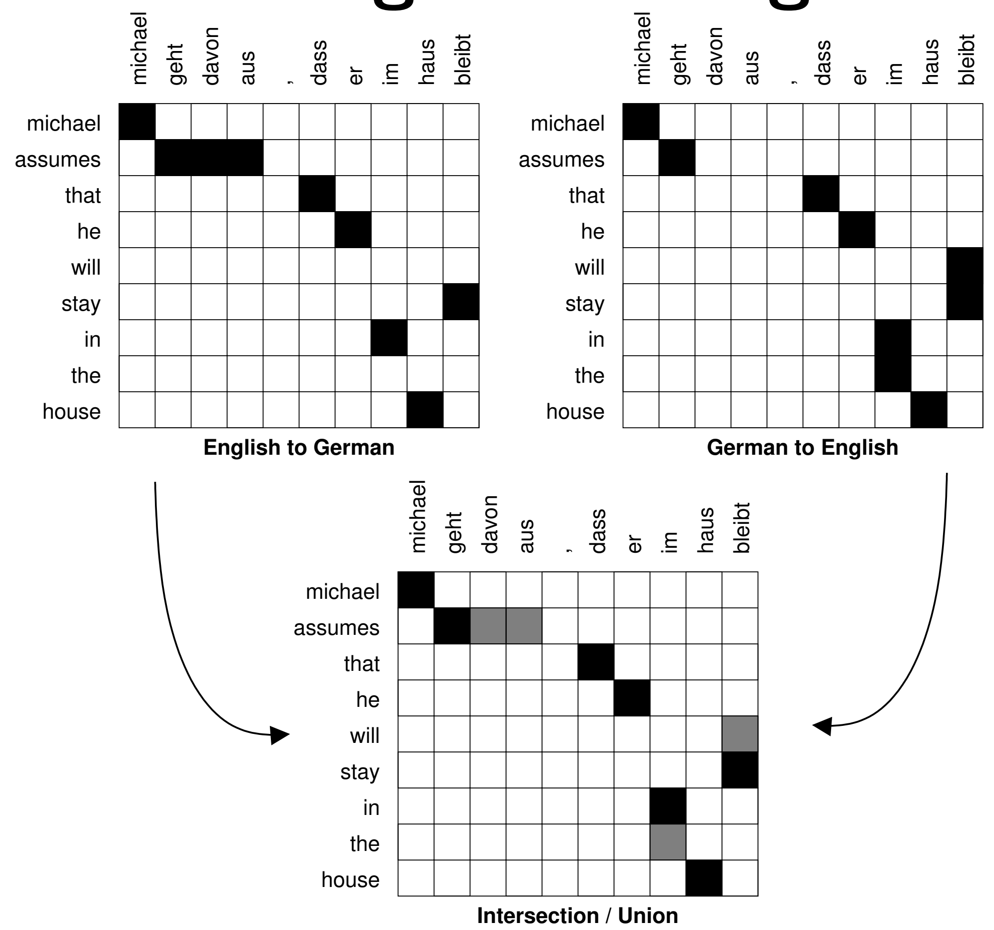
- However: different applications require different precision/recall trade-offs

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

Symmetrizing Word Alignments



- Intersection of GIZA++ bidirectional alignments
- Grow additional alignment points [Och and Ney, CompLing2003]

Growing heuristic

grow-diag-final(e2f,f2e)

```
neighboring = {(-1,0),(0,-1),(1,0),(0,1),(-1,-1),(-1,1),(1,-1),(1,1)}
alignment A = intersect(e2f,f2e); grow-diag(); final(e2f); final(f2e);
```

grow-diag()

```
while new points added do
    for all English word e ∈ [1...e_n], foreign word f ∈ [1...f_n], (e, f) ∈ A do
        for all neighboring alignment points (e_new, f_new) do
            if (e_new unaligned OR f_new unaligned) AND (e_new, f_new) ∈ union(e2f,f2e) then
                add (e_new, f_new) to A
            end if
        end for
    end for
end while
```

final()

```
for all English word e_new ∈ [1...e_n], foreign word f_new ∈ [1...f_n] do
    if (e_new unaligned OR f_new unaligned) AND (e_new, f_new) ∈ union(e2f,f2e) then
        add (e_new, f_new) to A
    end if
end for
```

More Recent Work on Symmetrization

- Symmetrize after each iteration of IBM Models [Matusov et al., 2004]
  - run one iteration of E-step for each direction
  - symmetrize the two directions
  - count collection (M-step)

- Use of posterior probabilities in symmetrization
  - generate n-best alignments for each direction
  - calculate how often an alignment point occurs in these alignments
  - use this posterior probability during symmetrization

Link Deletion / Addition Models

- Link deletion [Fossum et al., 2008]
  - start with union of IBM Model alignment points
  - delete one alignment point at a time
  - uses a neural network classifiers that also considers aspects such as how useful the alignment is for learning translation rules

- Link addition [Ren et al., 2007] [Ma et al., 2008]
  - possibly start with a skeleton of highly likely alignment points
  - add one alignment point at a time

Discriminative Training Methods

- Given some annotated training data, supervised learning methods are possible
- Structured prediction
  - not just a classification problem
  - solution structure has to be constructed in steps
- Many approaches: maximum entropy, neural networks, support vector machines, conditional random fields, MIRA, ...
- Small labeled corpus may be used for parameter tuning of unsupervised aligner [Fraser and Marcu, 2007]

Better Generative Models

- Aligning phrases
  - joint model [Marcu and Wong, 2002]
  - problem: EM algorithm likes really long phrases

- Fraser's LEAF
  - decomposes word alignment into many steps
  - similar in spirit to IBM Models
  - includes step for grouping into phrase

Summary

- Lexical translation
- Alignment
- Expectation Maximization (EM) Algorithm
- Noisy Channel Model
- 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
- Word Alignment