Lecture 4.3: Learning lexical translations

IBM model 1

Preparation for the lab

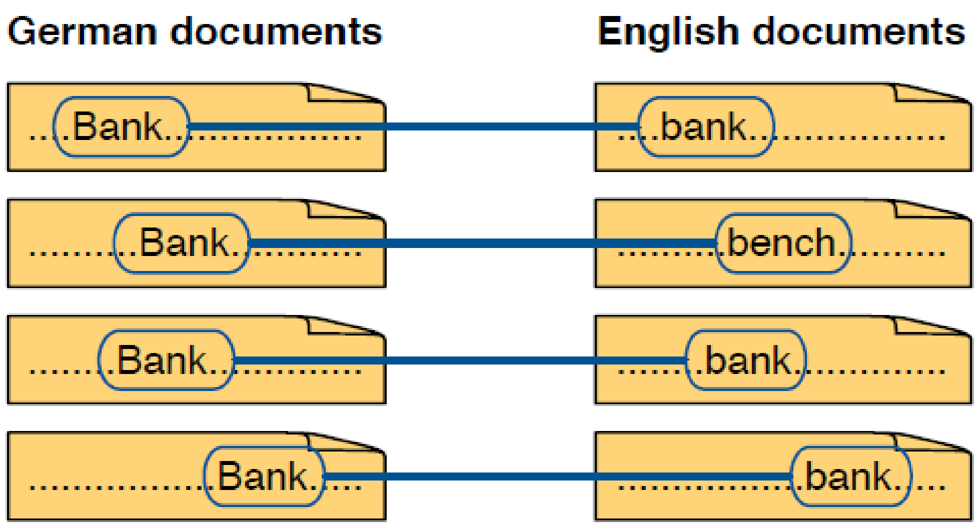
slides borrowed from Philipp Koehn

Guessing game:

Indonesian	English	
Ayam bakar	Grilled chicken	
Ayam goreng	Fried chicken	
Bebek goreng	Fried duck	
Ikan bakar	Grilled fish	
Ikan goreng	? ?	

Goreng=?
Ayam=?
Bakar=?
Ikan=?
Bebek=?

- Collect counts, infer probabilities

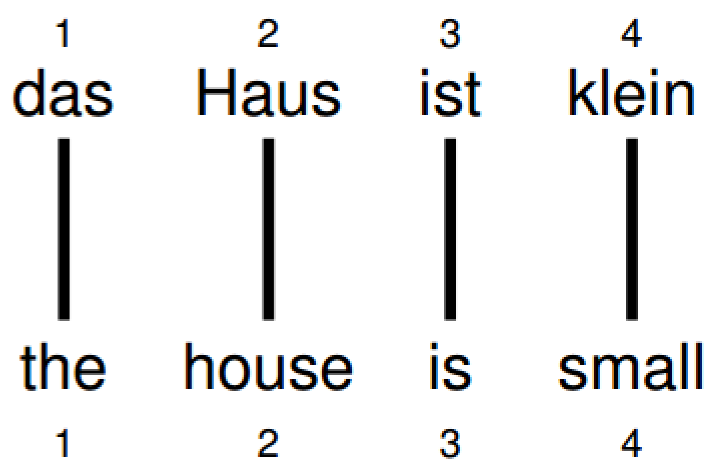


$\Rightarrow$ p(bank|Bank) = 0.75, p(bench|Bank) = 0.25

Courtesy of Philipp Koehn, Professor at The Johns Hopkins University

Alignments

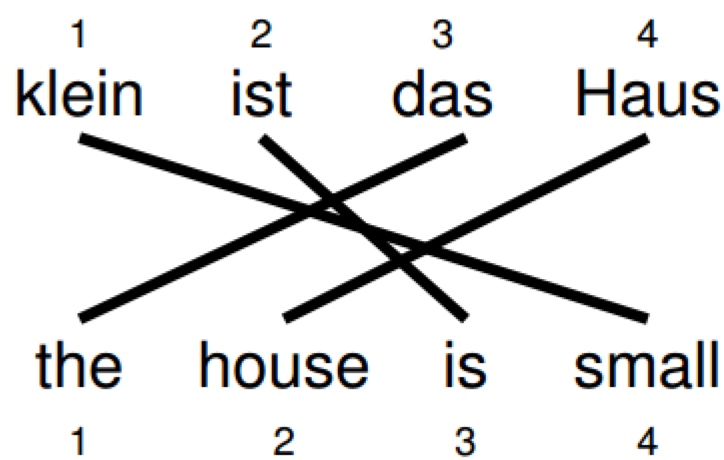
- Alignments are the “links” between words in parallel sentences
- Word positions numbered 1-4



English target word at position *i* to a German source word at position *j* with a function *a* : *i* → *j*

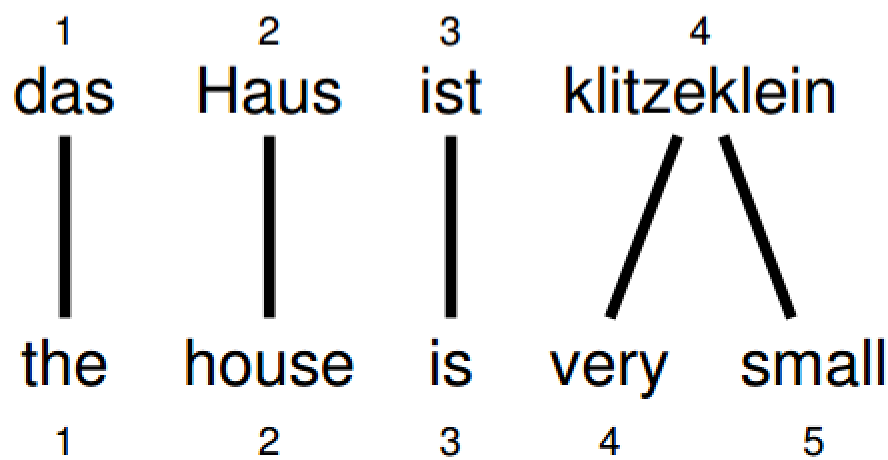
$$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4\}$$

Alignment reordering



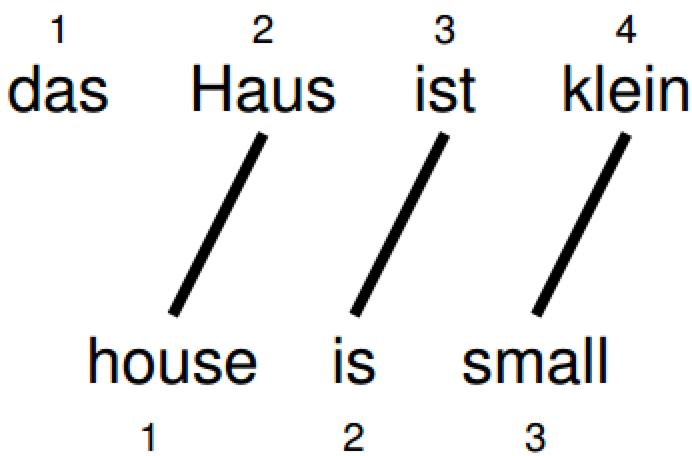
$$a : \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\}$$

Alignment: one to many

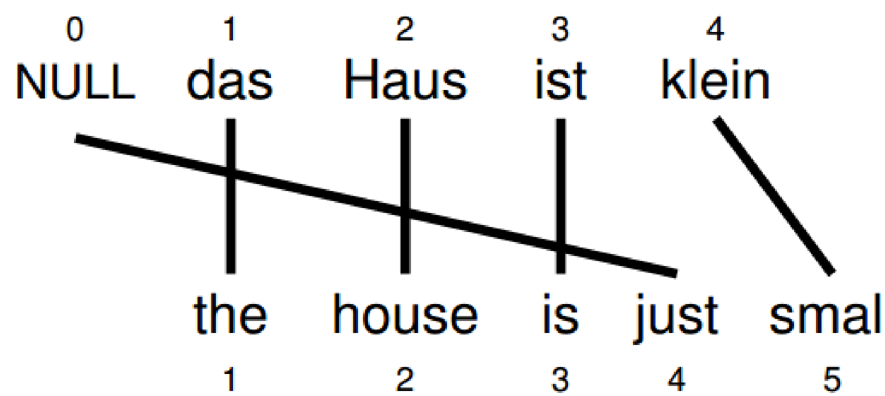


$$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4, 5 \rightarrow 4\}$$

Dropping/insert words



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



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

IBM model 1

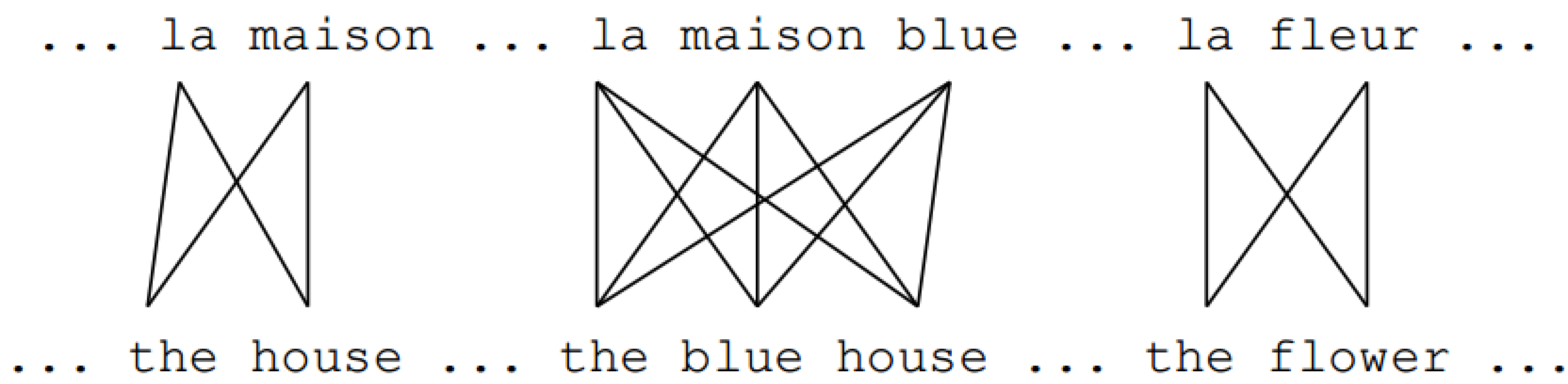
- Generative model: break up translation process into smaller steps

  IBM Model 1 only uses lexical translation
- Translation probability
  - for a foreign sentence $f = (f_1, \ldots, f_{lf})$ of length *lf*
  - to an English sentence $e = (e_1, \ldots, e_{le})$ of length *le*
  - with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \rightarrow i$

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

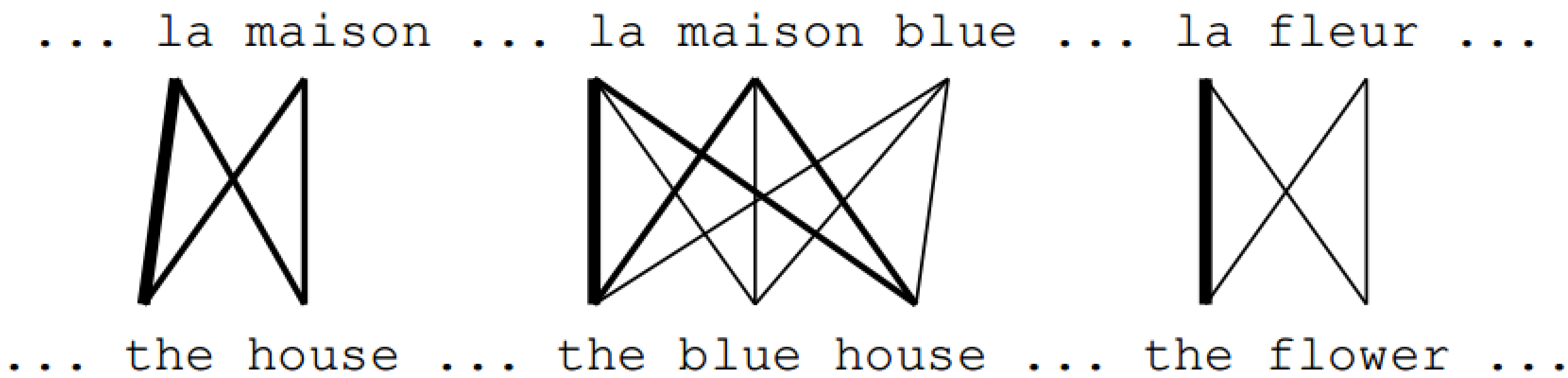
- parameter ε is a normalization constant

- Incomplete data
  – if we had complete data, we 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

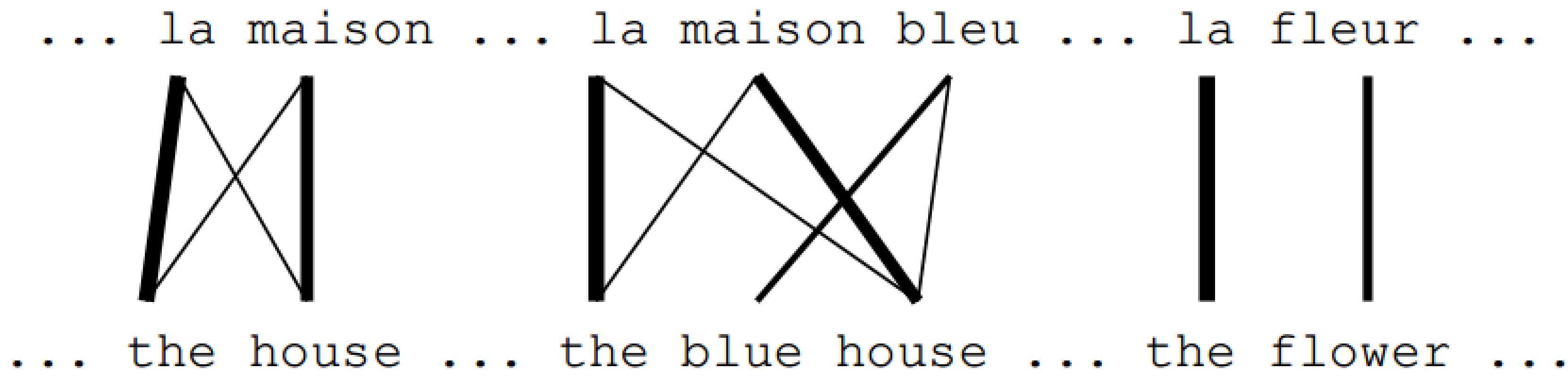


- Initial step: all alignments equally likely

- Model learns that, e.g., la is often aligned with the



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



- After another iteration

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

- Convergence
- Inherent hidden structure revealed by EM

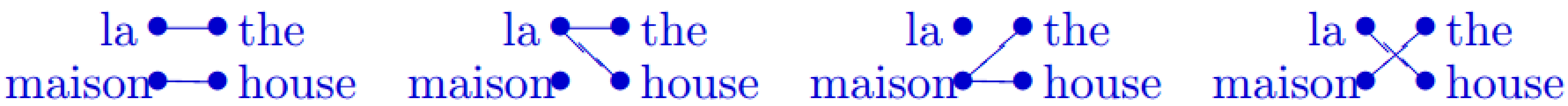

```
p(la|the) = 0.453
p(le|the) = 0.334
p(maison|house) = 0.876
p(bleu|blue) = 0.563
...
```

- Parameter estimation from the aligned corpus

- **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**



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

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

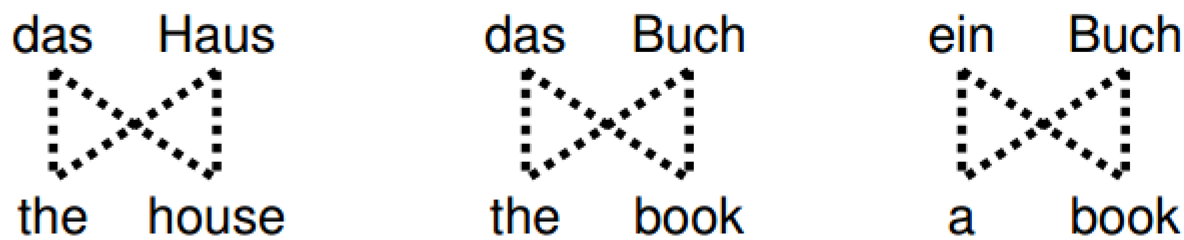
- **Counts**

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

IBM model 1 pseudo code

```
Input: set of sentence pairs (e, 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) += t(e|f) / s-total(e)
                total(f) += t(e|f) / 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) = count(e|f) / 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