Statistical Machine Translation
Lecture 2

Word Alignment Models

Stephen Clark

(based on slides by Philipp Koehn)

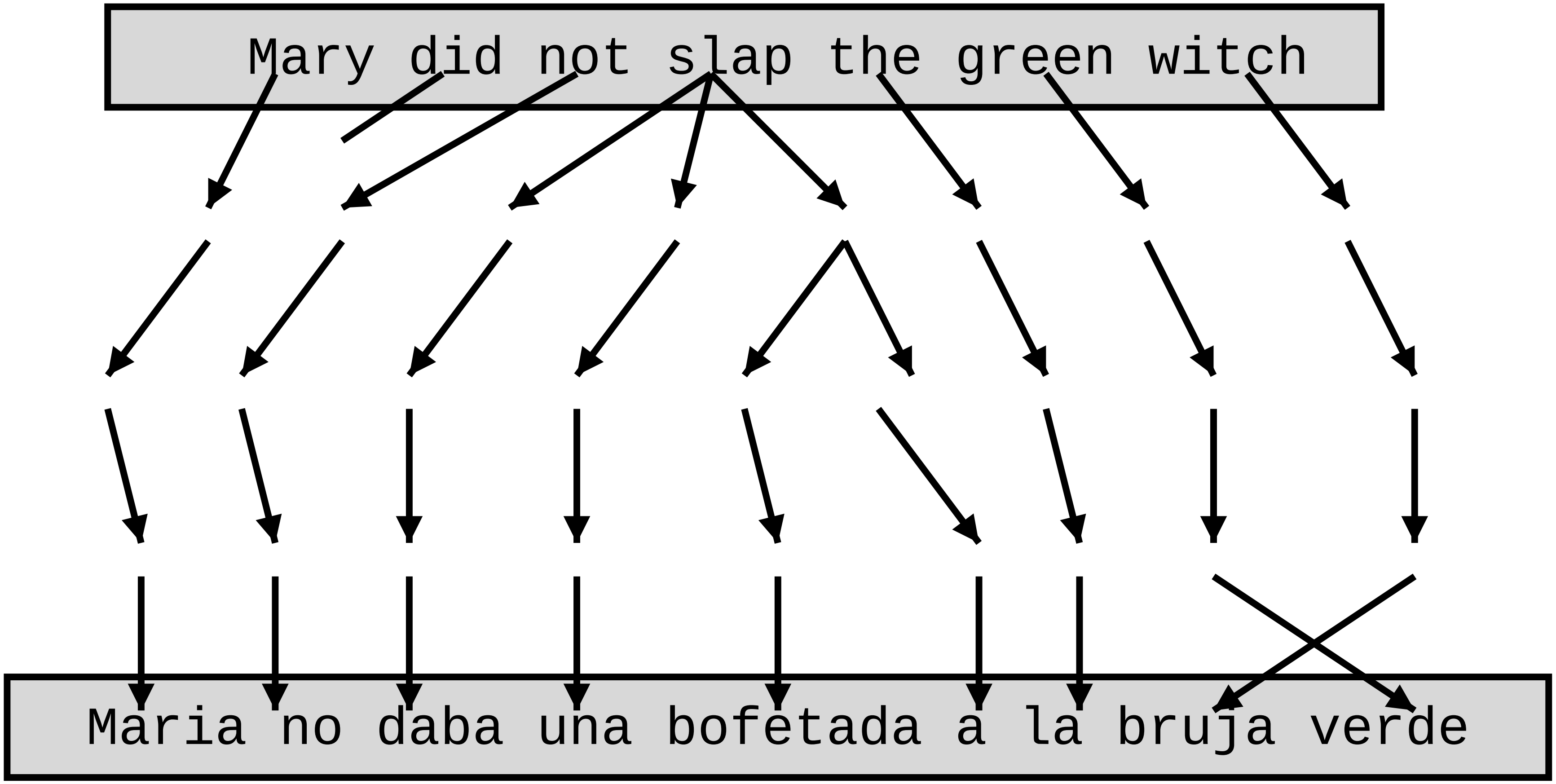
Statistical Modeling

`Mary did not slap the green witch`

`Maria no daba una bofetada a la bruja verde`

- Learn $P(f|e)$ from a parallel corpus
- Not sufficient data to estimate $P(f|e)$ directly

Statistical Modeling (2)



- Break the process into smaller steps

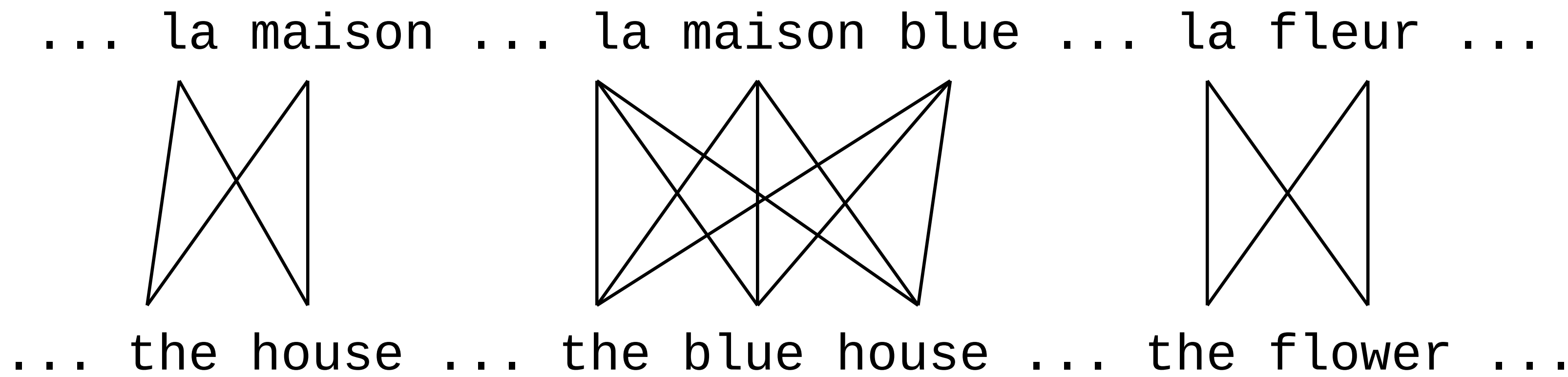
Statistical Modeling (3)

- Probabilities for smaller steps can be learned

Statistical Modeling (4)

- Generate a story how an English string e gets to be a foreign string f
  - choices in story are decided by reference to parameters
  - e.g., $p(\text{bruja}|\text{witch})$
- Formula for $P(f|e)$ in terms of parameters
  - usually long and hairy, but mechanical to extract from the story
- Training to obtain parameter estimates from incomplete data
  - Expectation Maximisation (EM) algorithm

Parallel Corpora

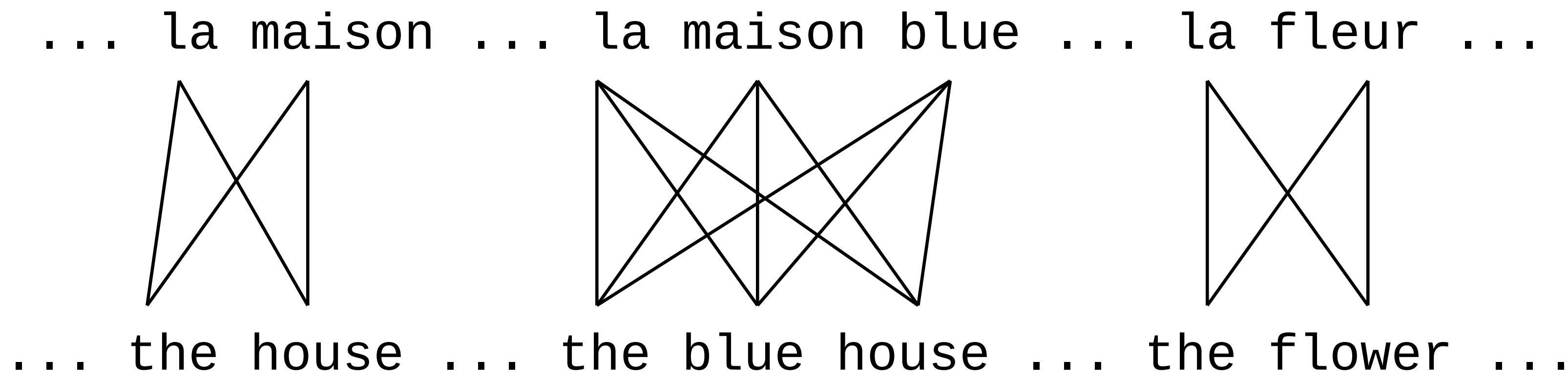


- Incomplete data
  - English and foreign words, but no connections between them
- Chicken and egg problem
  - if we had the connections, we could estimate the parameters of our generative story
  - if we had the parameters, we could estimate the connections

EM Algorithm

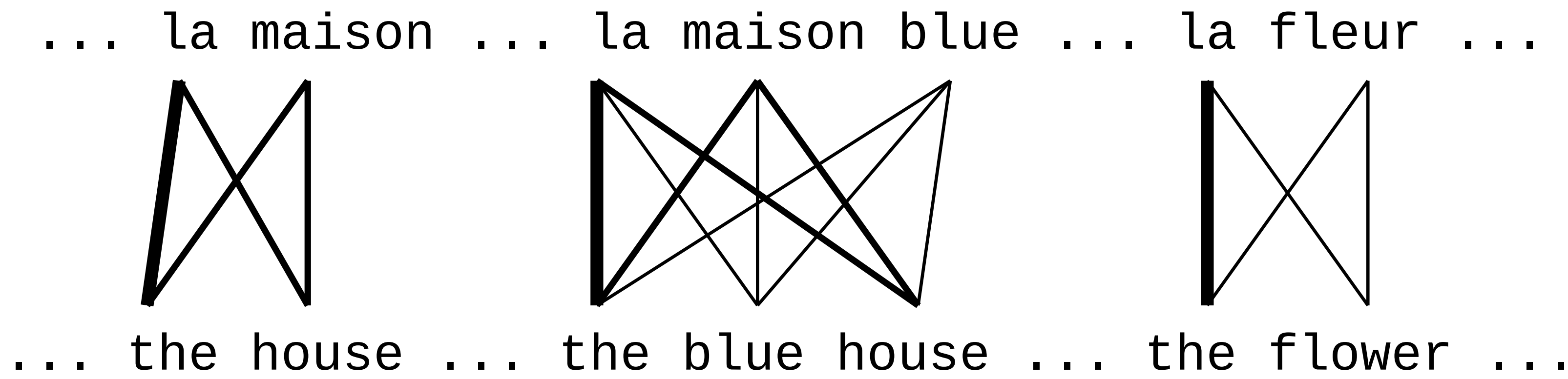
- Incomplete data
  - if we had complete data, we could estimate model
  - if we had model, we could fill in the gaps in the data
- EM in a nutshell
  - initialize model parameters (e.g. uniform)
  - assign probabilities to the missing data
  - estimate model parameters from completed data
  - iterate

EM Algorithm (2)



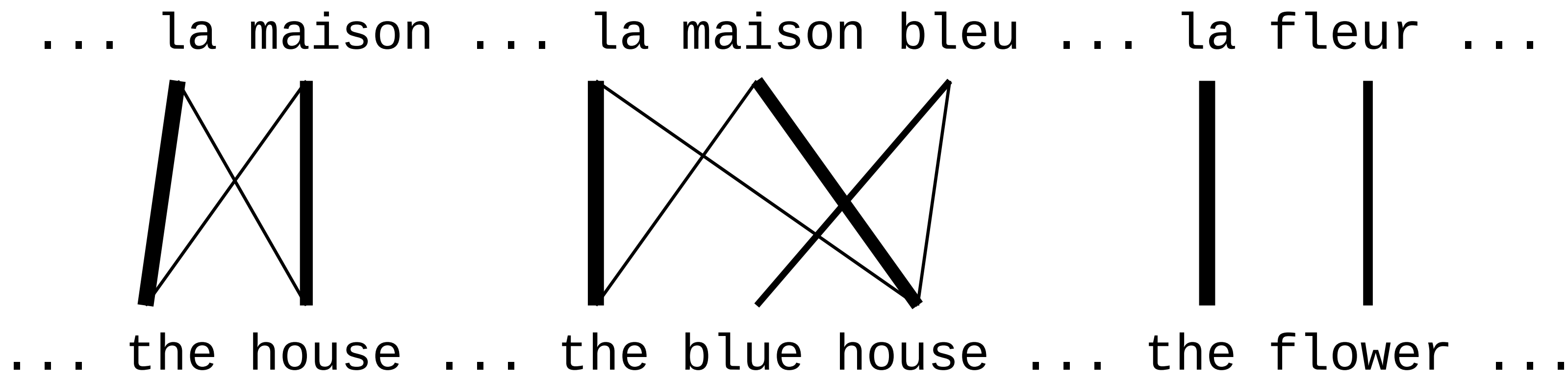
- Initial step: all connections equally likely
- Model learns that, e.g., la is often connected with the

EM Algorithm (3)



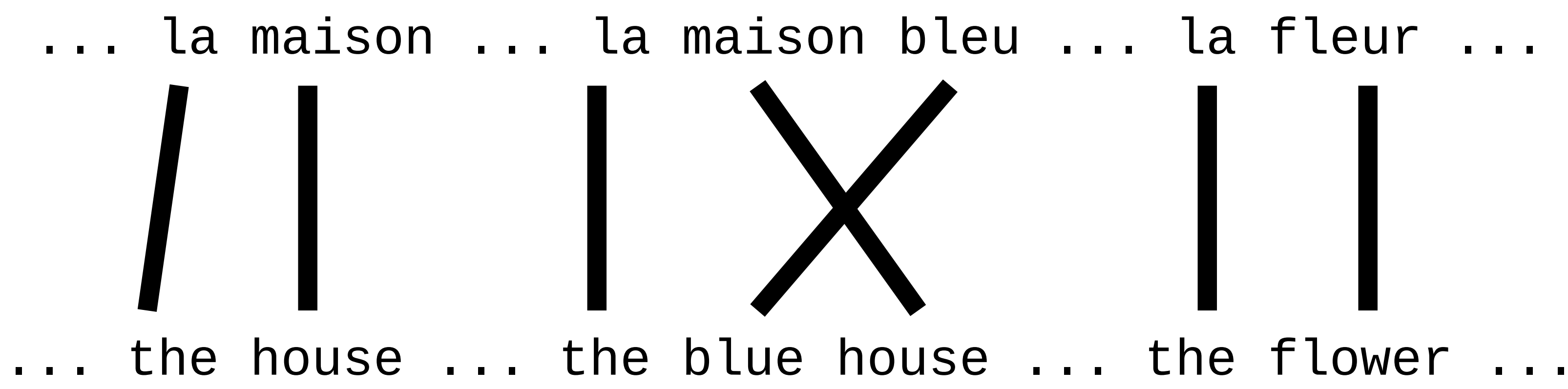
- After one iteration
- Connections, e.g., between la and the are more likely

EM Algorithm (4)



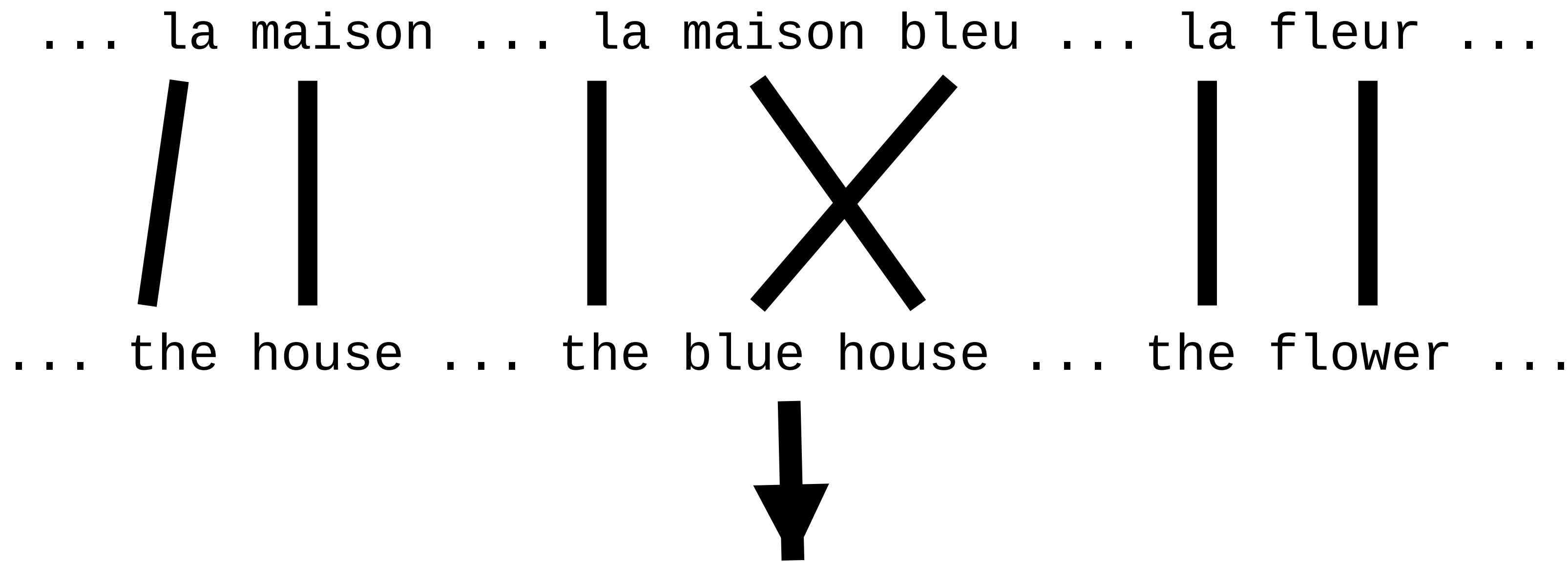
- After another iteration
- It becomes apparent that connections, e.g., between fleur and flower are more likely (pigeon hole principle)

EM Algorithm (5)



- Convergence
- Inherent hidden structure revealed by EM

EM Algorithm (6)




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

- Parameter estimation from the connected corpus

IBM Translation Models 1-5

- Choose a length for the French string, assuming all lengths to be equally likely (Models 1 and 2)
- For each position in the French string, connect it to the English and decide what French word to place there
  - Model 1 assumes all connections equally likely (so the order of the words in e and f has no impact (!))
  - Model 2 assumes the probability of a connection depends on the positions it connects and the lengths of the strings

IBM Translation Models 1-5

- In models 3, 4 and 5 we choose the number of words in f that connect to a particular English word, and then generate the French words
- In model 4 the probability of a connection depends in addition on the identites of the French and English words connected
- Models 3 and 4 are *deficient* - Model 5 is like Model 4 except it is not deficient
  - Models 3 and 4 waste probability mass on objects that aren't French strings at all

IBM Translation Models 1-5

- Why bother with all these models?
  - in particular why not just use Model 5, which makes less simplifying assumptions
- Models 1-4 serve as stepping stones to model 5
- Models 1 and 2 have a simple mathematical form so that iterations of EM can be performed exactly
  - can perform sums over all possible alignments
- Also Model 1 has a unique maximum so can use Model 1 to provide initial estimates for future models

Fundamental Equation

[add equation here]

IBM Model 1

$$p(\mathbf{f},\mathbf{a}|\mathbf{e}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} t(f_j|e_{a(j)})$$

- What is going on?
  - foreign sentence **f** = $f_1 ... f_m$
  - English sentence **e** = $e_1 ... e_l$
  - each French word f_j is generated by an English word $e_{a(j)}$, as defined by the alignment function a, with the probabilty t
  - $\epsilon = P(m|e)$ (can think of this as a constant normalisation factor)

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: Expectation Step

- Need the expected number of times word e connects to word f in translation $(\mathbf{f}|\mathbf{e})$
  - this is the (expected) *count* of f given e for $(\mathbf{f}|\mathbf{e})$

$$c(f|e;\mathbf{f},\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{e},\mathbf{f}) \sum_{j=1}^{m} \delta(f,f_j)\delta(e,e_{a(j)})$$

- Sum with double delta is just a fancy way of denoting the number of times e connects to f in $\mathbf{a}$

IBM Model 1 and EM: Expectation Step

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

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

- We already have the formula for $p(\mathbf{f}, \mathbf{a}|\mathbf{e})$ (definition of Model 1)

IBM Model 1 and EM: Expectation Step

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

$$
\begin{aligned}
p(\mathbf{f}|\mathbf{e}) &= \sum_{\mathbf{a}} p(\mathbf{f},\mathbf{a}|\mathbf{e}) \\
&= \sum_{a_1=0}^{l} \dots \sum_{a_m=0}^{l} p(\mathbf{f},\mathbf{a}|\mathbf{e}) \\
&= \sum_{a_1=0}^{l} \dots \sum_{a_m=0}^{l} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} t(f_j|e_{a(j)}) \\
&= \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \dots \sum_{a_m=0}^{l} \prod_{j=1}^{m} t(f_j|e_{a(j)}) \\
&= \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j|e_i)
\end{aligned}
$$

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

  $\longrightarrow$ this makes IBM Model 1 estimation tractable

IBM Model 1 and EM: Expectation Step

- Combine what we have:

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

$$= \frac{\frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m} t(f_j|e_{a(j)})}{\frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l} t(f_j|e_i)}$$

$$= \frac{\prod_{j=1}^{m} t(f_j|e_{a(j)})}{\prod_{j=1}^{m}\sum_{i=0}^{l} t(f_j|e_i)}$$

$$= \prod_{j=1}^{m} \frac{t(f_j|e_{a(j)})}{\sum_{i=0}^{l} t(f_j|e_i)}$$

IBM Model 1 and EM: Maximization Step

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

$$c(f|e;\mathbf{e},\mathbf{f}) = \sum_{\mathbf{a}} p(\mathbf{a}|\mathbf{e},\mathbf{f}) \sum_{j=1}^{m} \delta(f,f_j)\delta(e,e_{a(j)})$$

- With the same simplication as before:

$$c(f|e;\mathbf{e},\mathbf{f}) = \frac{t(f|e)}{\sum_{i=0}^{l} t(f|e_i)} \sum_{j=1}^{m} \delta(f,f_j) \sum_{i=0}^{l} \delta(e,e_i)$$

IBM Model 1 and EM: Maximization Step

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

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

IBM Model 1 and EM: Pseudocode

```
initialize t(f|e) uniformly
do
  set count(f|e) to 0 for all f,e
  set total(e) to 0 for all e
  for all sentence pairs (f_s,e_s)
    for all unique words f in f_s
      n_f = count of f in f_s
      total_s = 0
      for all unique words e in e_s
        total_s += t(f|e) * n_f
      for all unique words e in e_s
        n_e = count of e in e_s
        count(f|e) += t(f|e) * n_f * n_e / total_s
        total(e)   += t(f|e) * n_f * n_e / total_s
  for all e in domain( total(.) )
    for all f in domain( count(.|e) )
      t(f|e) = count(f|e) / total(e)
until convergence
```

Notes on IBM Model 1

- Model 1 in a nutshell: see how many times f and e appear together in the same sentence!
- So why bother with all this formalisation?
  - allows us to make our assumptions explicit
  - we can build on this simple model by relaxing some of the assumptions, and extending the mathematics
- Final parameter estimates do not depend on the initial assignments
  - likelihood function has a single maximum in this case
- Estimates from Model 1 can be used to initialise Model 2

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

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