# Statistical Machine Translation

Marcello Federico
FBK-irst Trento, Italy
Galileo Galilei PhD School - University of Pisa

Pisa, 7-19 May 2008





# Part II: Word Alignment Models

- Concept of Alignment
- Simple Alignment Model
- Parameter Estimation (EM algorithm)
- Fertility Alignment Models
- Hidden Markov Alignment Models

# Concept of Alignment

Given a translation (pair) $(\mathbf{f}, \mathbf{e})$ we introduce the concept of alignment which defines the **correspondence between single words** across the two sentences.

Let $\mathbf{f} = f_1, \ldots, f_j \ldots, f_m$ and $\mathbf{e} = e_1, \ldots, e_i \ldots, e_l$ we define an alignment $\mathcal{A}$ as a relation:

$$\mathcal{A} \subseteq \{(j,i) : j = 1, \ldots, m; i = 1, \ldots, l\}$$

- the concept of alignment reflects an intuitive idea of word correspondence
- however, sometimes the meaning of alignment becomes rather vague, e.g.:
  - idiomatic expressions
  - free translations
  - missing function words

In the following we show how alignments can be represented graphically.





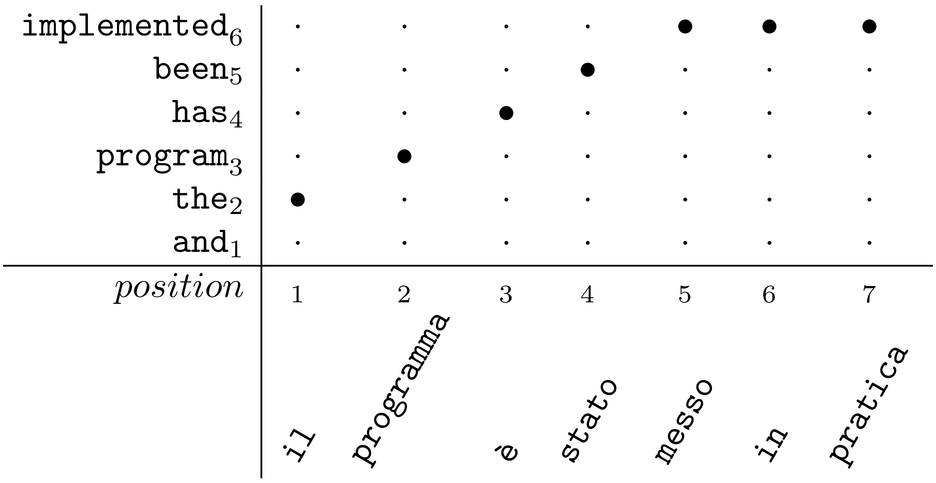
# Example 1: general alignment.

$$\mathcal{A} \subseteq \{(j,i) : j = 1, \ldots, m; i = 1, \ldots, l\}$$

| | | | | |
|---|---|---|---|---|
| $\texttt{money}_6$ | · | · | ● | ● |
| $\texttt{any}_5$ | · | · | ● | ● |
| $\texttt{have}_4$ | · | · | ● | ● |
| $\texttt{don't}_3$ | · | · | ● | ● |
| $\texttt{poor}_2$ | · | ● | · | · |
| $\texttt{the}_1$ | ● | · | · | · |
| *position* | 1 | 2 | 3 | 4 |
| | i | poveri | sono | nullatenenti |

# Example 2: Direct Alignment

$$\mathcal{A} : \{1, \ldots, m\} \longrightarrow \{1, \ldots, l\}$$

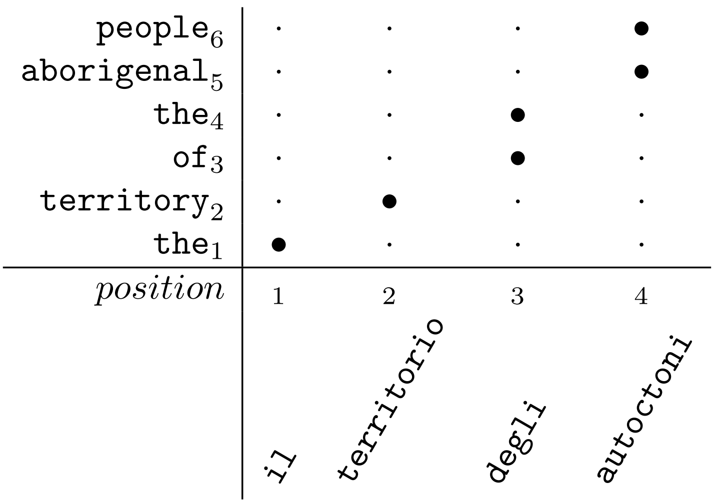







# Example 3: Inverted Alignment

$$\mathcal{A} : \{1, \ldots, l\} \longrightarrow \{1, \ldots, m\}$$



# Alignment

- Modelling the alignment as an **arbitrary relation** between source and target language is very general but **computationally unfeasible**: $2^{l \cdot m}$ possible alignments!
- A generally applied restriction is to let each source word be assigned to exactly one target word(see Example 2). Hence, alignment is a **map from source to target positions**:

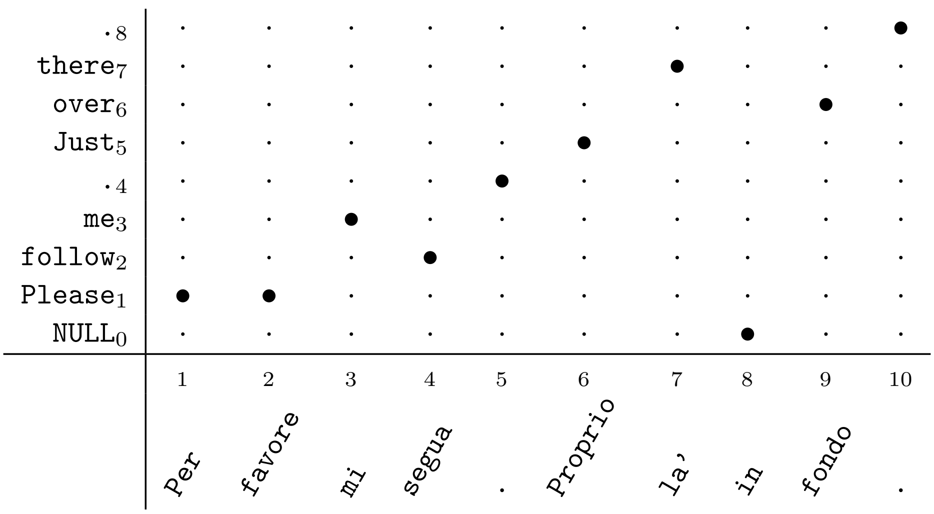
$$\mathcal{A} : \{1, \ldots, m\} \longrightarrow \{0, \ldots, l\}$$

- **Alignment variable**: $\mathbf{a} = a_1, \ldots, a_m$ consists of associations $j \to i = a_j$, from source position $j$ to target position $i = a_j$.
- We may include **null word alignments**, that is $a_j = 0$ to account for source words not aligned to any target word. Hence, "only" $(l + 1)^m$ possible alignments.





# Example: Non Monotone Alignment




[Exercise 2. Write the corresponding alignment $\mathbf{a}$.]

# Alignment Model

In SMT we will model the **translation probability** $\Pr(\mathbf{f} \mid \mathbf{e})$ by summing the probabilities of all possible $(l+1)^m$ 'hidden' alignments $\mathbf{a}$ between the source and the target strings:

$$\Pr(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a}} \Pr(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) \tag{1}$$

Hence we will consider **statistical alignment models**:

$$\Pr(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$$

defined by specific sets of parameters $\theta$.

The art of statistical modelling consists in designing statistical models which **capture the relevant properties of the considered phenomenon**, in our case the relationship between a source language string and a target language string.





# Often used symbols

| | |
|---|---|
| $l, m$ | length of target and source sentences |
| $\mathbf{f} = f_1^m \equiv f_1 \ldots, f_m$ | source sentence |
| $\mathbf{e} = e_1^l \equiv e_1 \ldots, e_l$ | target sentence |
| $i, j$ | target and source positions |
| $e_i, f_j$ | target and source words |
| $e_0$ | empty word (of the target sentence) |
| $i \in \{0, 1, \ldots, l\}$ | target positions |

# Alignment Model

One of the many ways to exactly decompose $\Pr(f_1^m, a_1^m \mid e_1^l)$ is:

$$\begin{aligned}\Pr(f_1^m, a_1^m \mid e_1^l) &= \Pr(m \mid e_1^l) \prod_{j=1}^{m} \Pr(f_j, a_j \mid f_1^{j-1}, a_1^{j-1}, m, e_1^l) \\ &= \Pr(m \mid e_1^l) \prod_{j=1}^{m} \Pr(a_j \mid f_1^{j-1}, a_1^{j-1}, m, e_1^l) \cdot \Pr(f_j \mid f_1^{j-1}, a_1^{j}, m, e_1^l)\end{aligned}$$

**Generative stochastic process**:

1. choose length $m$ of the French string, given knowledge of the English string $\mathbf{e}_1^l$
2. cover one English position for each French position $j$, given ...
3. choose French word for each position $j$ , given the covered English position ....





# IBM Model 1

Given the general alignment model:

$$\Pr(f_1^m, a_1^m \mid e_1^l) = \Pr(m \mid e_1^l) \prod_{j=1}^{m} \Pr(a_j \mid f_1^{j-1}, a_1^{j-1}, m, e_1^l) \cdot \Pr(f_j \mid f_1^{j-1}, a_1^{j}, m, e_1^l)$$

We **simplify all interactions** by means of pairwise dependencies:

$$\begin{aligned}\Pr(m \mid e_1^l) &= p(m \mid l) && \text{string length model} \\ \Pr(a_j \mid f_1^{j-1}, a_1^{j-1}, m, e_1^l) &= (l+1)^{-1} && \text{alignment probabilities} \\ \Pr(f_j \mid f_1^{j-1}, a_1^{j}, m, e_1^l) &= p(f_j \mid e_{a_j}) && \text{translation probabilities}\end{aligned}$$

Hence, we get the following translation model:

$$\Pr(f_1^m \mid e_1^l) = \sum_{a_1^m} \Pr(f_1^m, a_1^m \mid e_1^l) = p(m \mid l)(l+1)^{-m} \cdot \sum_{a_1^m} \prod_{j=1}^{m} p(f_j \mid e_{a_j})$$

# IBM Model 1

Model 1 corresponds to the following **stochastic generative process**:

1. Choose a length $m$ for $\mathbf{f}$ according to $p(m \mid l)$
2. For each $j = 1, \ldots, m$, choose $a_j$ in $\{0, 1, \ldots, l\}$ at random
3. For each $j = 1, \ldots, m$, choose French word $f_j$ according to $p(f_j \mid e_{a_j})$

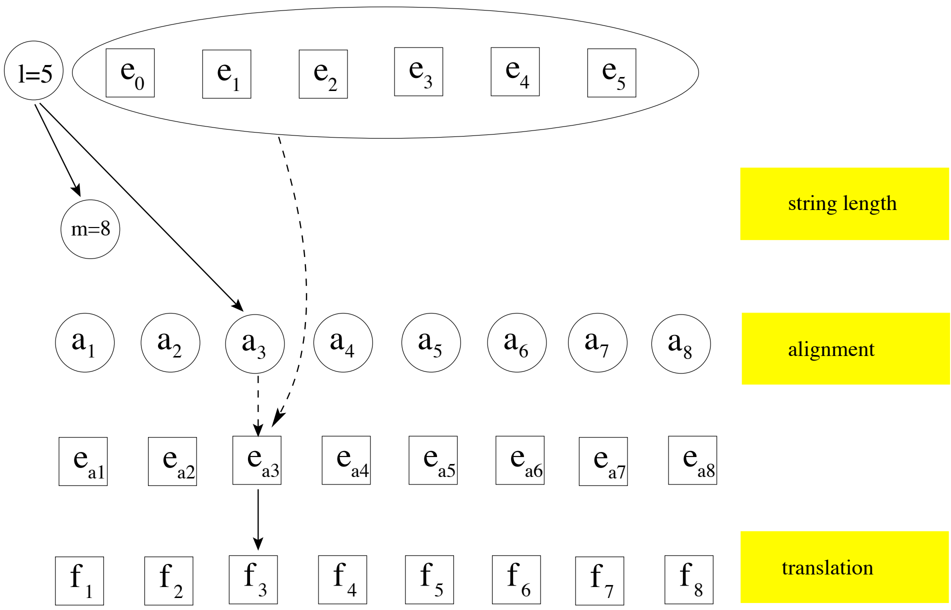
**Properties**:

- Model 1 is very naive but is a good starting point for better models
- Training of Model 1, that is estimating its probability tables, is very efficient
- Training can exploit a parallel corpus without alignments





# IBM Model 1: Dependency Diagram



# IBM Model 2

- Replaces the uniform alignment probability of Model 1 with:

$$\Pr(a_j \mid f_1^{j-1}, a_1^{j-1}, m, e_1^l) = p(a_j \mid j, l, m)$$

- **Properties**:
  - Model 1 does not care were words appear in the two strings!
  - Model 2 introduces alignment probs, i.e. a table of size $(l_{max} \times m_{max})^2$
  - Training of Models 1-2 is easy from a bilingual corpus with given alignments:
    - both models are a product of discrete distributions
    - their likelihood function is a product of multinomial distributions
    - ML estimation just needs relative counts of events (sufficient statistics)



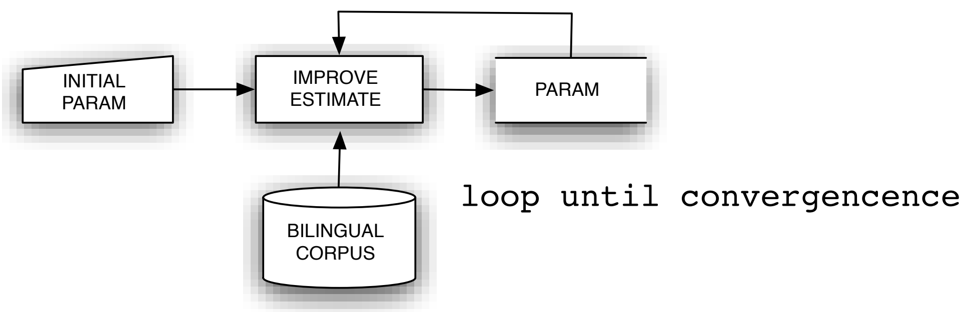


# Estimation of IBM Models

## How to train alignment models?

- we could use alignments to train the model (MLE)
- we could compute the alignments through the model (Viterbi alignment)

Idea to solve this chicken & egg problem:



# Training of Alignment Models

Given a translation model $p_\theta(\mathbf{f} \mid \mathbf{e})$, unknown parameters $\theta$ can be estimated with a large parallel corpus by applying the **maximum likelihood criterion**.

Given a sample of translations $\{(\mathbf{f}_s, \mathbf{e}_s) : s = 1, \ldots, S\}$, we maximize:

$$\psi(\theta) = S^{-1} \sum_{s=1}^{S} \log p_\theta(\mathbf{f}_s \mid \mathbf{e}_s) = \sum_{\mathbf{f},\mathbf{e}} C(\mathbf{f}, \mathbf{e}) \log p_\theta(\mathbf{f} \mid \mathbf{e}) \quad (2)$$

where $C(\mathbf{f}, \mathbf{e})$ is $1/S$ times the number of times $(\mathbf{f}, \mathbf{e})$ occurs in the sample.

We have seen that $p_\theta(\mathbf{f}_s \mid \mathbf{e}_s)$ is the marginal probability of an alignment model:

$$p_\theta(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a}} p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) \quad (3)$$

We devise an algorithm which modifies $\theta$ to increase the likelihood $\psi(\theta)$.





# Relative Objective Function

We compare alignment models $p_{\tilde{\theta}}$ and $p_\theta$ using the relative objective function:

$$R(\tilde{\theta}, \theta) = \sum_{\mathbf{f},\mathbf{e}} C(\mathbf{f}, \mathbf{e}) \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) \log \frac{p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{f}, \mathbf{a} \mid \mathbf{e})} \quad (4)$$

where:

$$p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) = \frac{p_{\tilde{\theta}}(\mathbf{f}, \mathbf{a} \mid \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{f} \mid \mathbf{e})} = \frac{p_{\tilde{\theta}}(\mathbf{f}, \mathbf{a} \mid \mathbf{e})}{\sum_{\mathbf{a}'} p_{\tilde{\theta}}(\mathbf{f}, \mathbf{a}' \mid \mathbf{e})} \quad (5)$$

and

$$R(\tilde{\theta}, \tilde{\theta}) = 0 \quad (6)$$

**Function $R$ is important for its properties, don't put effort to interpret it.**

We can introduce now the Expectation-Maximization Algorithm.

# EM Theorem

**Theorem**. Given models $p_{\tilde{\theta}}$ and $p_\theta$, it holds:

$$\text{if } R(\tilde{\theta}, \theta) > 0 \text{ then } \psi(\theta) > \psi(\tilde{\theta}) \tag{7}$$

**Proof** We can show that $R$ is related to the likelihood function $\psi$ by:

$$\psi(\theta) \geq \psi(\tilde{\theta}) + R(\tilde{\theta}, \theta) \tag{8}$$

which is is equivalent to the theorem's statement. The proof of the inequality is based on this simple geometric property:

$$\log x \leq (x - 1), \text{ with equality holding if } x = 1$$





# Proof of EM Theorem

Hence, for any $\mathbf{e}$ and $\mathbf{f}$, we have that

$$\sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) \log \frac{p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{f}, \mathbf{a} \mid \mathbf{e})} \tag{9}$$

$$= \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) \log \left( \frac{p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) / p_\theta(\mathbf{f} \mid \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) / \tilde{p}_\theta(\mathbf{f} \mid e)} \cdot \frac{p_\theta(\mathbf{f} \mid \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{f} \mid \mathbf{e})} \right) \tag{10}$$

$$= \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) \log \frac{p_\theta(\mathbf{a} \mid \mathbf{f}, \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e})} + \log \frac{p_\theta(\mathbf{f} \mid \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{f} \mid \mathbf{e})} \underbrace{\sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e})}_{=1} \tag{11}$$

$$\leq \underbrace{\sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) \left( \frac{p_\theta(\mathbf{a} \mid \mathbf{f}, \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e})} - 1 \right)}_{=0} + \log \frac{p_\theta(\mathbf{f} \mid \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{f} \mid \mathbf{e})} \tag{12}$$

# Proof of EM Theorem

By summing up over all $(\mathbf{f},\mathbf{e})$ we get the desired inequality:

$$\begin{aligned}
\sum_{(\mathbf{f},\mathbf{e})} C(\mathbf{f},\mathbf{e}) \log \frac{p_\theta(\mathbf{f} \mid \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{f} \mid \mathbf{e})} &\geq \sum_{(\mathbf{f},\mathbf{e})} C(\mathbf{f},\mathbf{e}) \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f},\mathbf{e}) \log \frac{p_\theta(\mathbf{f},\mathbf{a} \mid \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{f},\mathbf{a} \mid \mathbf{e})} \\
\psi(\theta) - \psi(\tilde{\theta}) &\geq R(\tilde{\theta},\theta) \\
\psi(\theta) &\geq \psi(\tilde{\theta}) + R(\tilde{\theta},\theta)
\end{aligned}$$

**End of Proof.**

- we now need to maximize $R$ in order to find better parameters
- ... but we need some parameters to start with (chicken-egg problem)
- the good news is that we can start with any settings (uniform, random, ...)
- the EM algorithm iterates the maximization of R





# EM Algorithm

The EM algorithm exploits an auxiliary function $R(\tilde{\theta},\theta)$ with the properties:

$$R(\theta,\theta) = 0 \text{ and } \psi(\theta) > \psi(\tilde{\theta}) \text{ if } R(\tilde{\theta},\theta) > 0.$$

The following **iterative procedure** is applied to find optimal parameter values:

0. Choose some initial values $\tilde{\theta}$
1. Repeat Steps 2-3 until convergence
2. With $\tilde{\theta}$ fixed, find values $\theta$ that maximize $R(\tilde{\theta},\theta)$
3. Replace $\tilde{\theta}$ by $\theta$

**Notice**: For any $\tilde{\theta}$, $max_\theta\, R(\tilde{\theta},\theta) \geq 0$, since at least $R = 0$ when $\tilde{\theta} = \theta$.

# Parameter Estimation with EM Algorithm

Let us consider a **simple alignment model**, simpler than Model 1 and Model 2:

$$p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = \prod_{\omega \in \Omega} \theta(\omega)^{c(\omega; \mathbf{a}, \mathbf{f}, \mathbf{e})} \tag{13}$$

where:

- $\theta(\omega)$ as the probability of event $\omega \in \Omega$
- $c(\omega; \mathbf{a}, \mathbf{f}, \mathbf{e})$ as the frequency of $\omega$ in $(\mathbf{a}, \mathbf{f}, \mathbf{e})$

hence, we require that:

$$\theta(\omega) \geq 0 \qquad \sum_{\omega \in \Omega} \theta(\omega) = 1 \tag{14}$$





# Parameter Estimation with EM Algorithm

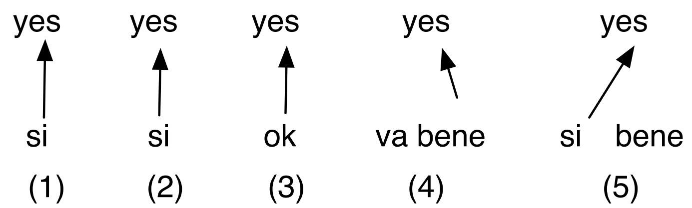
Simple example: learn the (single) Italian word for "yes":

$$p_\theta(\mathbf{f}, \mathbf{a} \mid yes) = \prod_{f \in \{si, ok, va, bene\}} p(f|yes)^{c(f; \mathbf{a}, \mathbf{f}, yes)} \tag{15}$$

where:

- $a_j = 0$ or $1$ and $\sum_j a_j = 1$
- $c(f; \mathbf{a}, \mathbf{f}, yes)$ is 1 if $f$ is aligned to "yes" and 0 otherwise

Training data with correct alignment (not given).



## Parameter Estimation with EM Algorithm

It is easy to verify that:

$$c(\omega; \mathbf{a}, \mathbf{f}, \mathbf{e}) = \theta(\omega) \frac{\partial}{\partial \theta(\omega)} \log p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) \tag{16}$$

From the concavity of the objective function $R(\tilde{\theta}, \theta)$, necessary and sufficient conditions to determine values for $\theta$ which maximize $R(\tilde{\theta}, \theta)$ are:

$$\begin{cases} \frac{\partial}{\partial \theta(\omega)} \left( R(\tilde{\theta}, \theta) + \lambda(1 - \sum_{\omega \in \Omega} \theta(\omega) \right) = \frac{\partial}{\partial \theta(\omega)} R(\tilde{\theta}, \theta) - \lambda = 0, \quad \omega \in \Omega \\ \frac{\partial}{\partial \lambda} \left( R(\tilde{\theta}, \theta) + \lambda(1 - \sum_{\omega \in \Omega} \theta(\omega) \right) = 1 - \sum_{\omega \in \Omega} \theta(\omega) = 0 \end{cases} \tag{17}$$

where $\lambda$ is a Lagrange multiplier for constraint (14).





## Parameter Estimation with EM Algorithm

From the previous result:

$$\begin{aligned} \frac{\partial}{\partial \theta(\omega)} R(\tilde{\theta}, \theta) &= \frac{\partial}{\partial \theta(\omega)} \sum_{\mathbf{f},\mathbf{e}} C(\mathbf{f}, \mathbf{e}) \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) \log \frac{p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e})}{p_{\tilde{\theta}}(\mathbf{f}, \mathbf{a} \mid \mathbf{e})} \\ &= \sum_{\mathbf{f},\mathbf{e}} C(\mathbf{f}, \mathbf{e}) \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) \frac{\partial}{\partial \theta(\omega)} \log p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) \\ &= \sum_{\mathbf{f},\mathbf{e}} C(\mathbf{f}, \mathbf{e}) \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) \frac{c(\omega; \mathbf{a}, \mathbf{f}, \mathbf{e})}{\theta(\omega)} \end{aligned} \tag{18}$$

This result can be plugged into the first equation of system (17).

# Parameter Re-estimation Formulae

By multiplying equation (17) by $\theta(\omega)$, taking the sum over all $\omega$ we get:

$$\sum_{\omega\in\Omega}\sum_{\mathbf{f},\mathbf{e}} C(\mathbf{f},\mathbf{e})\sum_{\mathbf{a}} p_{\tilde\theta}(\mathbf{a}\mid\mathbf{f},\mathbf{e})c(\omega;\mathbf{a},\mathbf{f},\mathbf{e}) - \lambda\underbrace{\sum_{\omega\in\Omega}\theta(\omega)}_{=1} = 0 \qquad (19)$$

From the solution for $\lambda$ and for each $\theta(\omega)$ we get the re-estimation formulae:

$$\theta(\omega) = c_{\tilde\theta}(\omega)/\lambda \qquad \lambda = \sum_{\omega\in\Omega} c_{\tilde\theta}(\omega) \qquad (20)$$

$$c_{\tilde\theta}(\omega) = \sum_{\mathbf{f},\mathbf{e}} C(\mathbf{f},\mathbf{e})\sum_{\mathbf{a}} p_{\tilde\theta}(\mathbf{a}\mid\mathbf{f},\mathbf{e})c(\omega;\mathbf{a},\mathbf{f},\mathbf{e}) \qquad (21)$$

- $\theta(\omega)$ is the expected relative frequency of $\omega$ according to $p_{\tilde\theta}(\mathbf{a}\mid\mathbf{f},\mathbf{e})$
- hence, the new estimate of the parameter correspond to a relative frequency ..
- computed with counters, as we had observed the alignments!
- the trick is to calculate the expected counts with the current model





# Extension to IBM Alignment Models

The previous result can be easily extended to **models of the general form** (15):

$$p_\theta(\mathbf{f},\mathbf{a}\mid\mathbf{e}) = \prod_{\omega\in\Omega}\theta(\omega)^{c(\omega;\mathbf{a},\mathbf{f},\mathbf{e})} \qquad (22)$$

for which the single constraint equation (14) is replaced by multiple constraints

$$\sum_{\omega\in\Omega_\mu}\theta(\omega) = 1, \qquad \mu = 1,2,\ldots \qquad (23)$$

where the subsets $\Omega_\mu$, $\mu = 1,2,\ldots$, form a partition of $\Omega$.

Important: **Model 1 and Model 2 can be represented in this way**.

## Extension to IBM Alignment Models

Constraints leads to the system of equations:

$$\begin{cases} \frac{\partial}{\partial\theta(\omega)}\left(R(\tilde{\theta},\theta)+\sum_\mu \lambda_\mu\left(1-\sum_{\omega\in\Omega_\mu}\theta(\omega)\right)\right) = \frac{\partial}{\partial\theta(\omega)}R(\tilde{\theta},\theta)-\lambda_\mu = 0 \\ \qquad \omega\in\Omega_\mu, \mu=1,2,\ldots \\ \frac{\partial}{\partial\lambda_\mu}\left(R(\tilde{\theta},\theta)+\lambda_\mu(1-\sum_{\omega\in\Omega\_mu}\theta(\omega)\right) = 1-\sum_{\omega\in\Omega_\mu}\theta(\omega)=0 \qquad \mu=1,2,\ldots \end{cases} \tag{24}$$

For $\omega \in \Omega_\mu$, after a similar procedure we get the **re-estimation formula**

$$\theta(\omega) = \lambda_\mu^{-1} c_{\tilde{\theta}}(\omega) \qquad \lambda_\mu = \sum_{\omega\in\Omega_\mu} c_{\tilde{\theta}}(\omega) \tag{25}$$

$$c_{\tilde{\theta}}(\omega) = \sum_{\mathbf{f},\mathbf{e}} C(\mathbf{f},\mathbf{e}) \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a}\mid\mathbf{f},\mathbf{e})c(\omega;\mathbf{a},\mathbf{f},\mathbf{e}) \tag{26}$$

Again, we use relative frequencies over expected counts!





## Training Model 2

For Model 2, $\Omega = \{(i,j,l,m)\} \cup \{(e,f)\}$, which is partitioned as follows:

$$\Omega_{j,l,m} = \{(i\mid j,l,m): 0\le i\le l\}, \quad 0\le j,m\le m_{max}, 0\le l\le l_{max}$$

$$\Omega_e = \{(f\mid e): f\in\mathcal{F}\}, \quad e\in\mathcal{E} \tag{27}$$

$$c(i\mid j,l,m;\mathbf{a},\mathbf{f},\mathbf{e}) = \delta(i,a_j) \tag{28}$$

$$c(f\mid e;\mathbf{a},\mathbf{f},\mathbf{e}) = \sum_{j=1}^{m}\delta(e,e_{a_j})\delta(f,f_j) \tag{29}$$

We can now directly derive the iterative re-estimation formulae from:

$$c_{\tilde{\theta}}(\omega;\mathbf{f},\mathbf{e}) = \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a}\mid\mathbf{f},\mathbf{e})c(\omega;\mathbf{a},\mathbf{f},\mathbf{e}) \tag{30}$$

**Problem**: the above formula requires summing over $(l+1)^m$ alignments!

# Training Model 2: Useful Formulas

**Model 2 permits to efficiently calculate the sum over alignments**:

$$p_\theta(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a}} p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = p(m \mid l) \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} p(f_j \mid e_{a_j}) p(a_j \mid j, l, m)$$
$$= p(m \mid l) \prod_{j=1}^{m} \sum_{i=0}^{l} p(f_j \mid e_i) p(i \mid j, l, m) \qquad (31)$$

Proof. Let $m = 3$ and $l = 1$, and let $x_{ja_j} \equiv p(f_j \mid e_{a_j}) p(a_j \mid j, l, m)$. It is routine to verify that: $x_{10}x_{20}x_{30} + \ldots + x_{11}x_{21}x_{30} + x_{11}x_{21}x_{31} = (x_{10} + x_{11})(x_{20} + x_{21})(x_{30} + x_{31})$
Hence we can write:

$$p_\theta(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) = \frac{p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e})}{\sum_{\mathbf{a}} p_\theta(\mathbf{f}, \mathbf{a} \mid \mathbf{e})} = \frac{\prod_{j=1}^{m} p(f_j \mid e_{a_j}) p(a_j \mid j, l, m)}{\prod_{j=1}^{m} \sum_{i=0}^{l} p(f_j \mid e_i) p(i \mid j, l, m)} \equiv \prod_{j=1}^{m} p_\theta(a_j \mid j, \mathbf{f}, \mathbf{e}) \qquad (32)$$

**Important**: we need only $2 \cdot m \cdot (l + 1)$ operations!





# Training Model 2

Now we are ready to write the re-estimation formulas from eq. (28-29) (32):

$$c_{\tilde\theta}(f \mid e; \mathbf{f}, \mathbf{e}) = \sum_{\mathbf{a}} p_{\tilde\theta}(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) c(f \mid e; \mathbf{a}, \mathbf{f}, \mathbf{e})$$
$$= \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \left( \prod_{j=1}^{m} p_{\tilde\theta}(a_j \mid j, \mathbf{f}, \mathbf{e}) \right) \sum_{k=1}^{m} \left( \delta(e, e_{a_k}) \delta(f, f_k) \right)$$
$$= \sum_{k=1}^{m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} p_{\tilde\theta}(a_j \mid j, \mathbf{f}, \mathbf{e}) \delta(e, e_{a_k}) \delta(f, f_k)$$
$$= \sum_{j=1}^{m} \sum_{i=0}^{l} p_{\tilde\theta}(i \mid j, \mathbf{f}, \mathbf{e}) \delta(e, e_i) \delta(f, f_j) \qquad (33)$$
$$= \sum_{j=1}^{m} \sum_{i=0}^{l} \frac{p(f_j \mid e_i) p(i \mid j, l, m)}{\sum_{i'=0}^{l} p(f_j \mid e_{i'}) p(i' \mid j, l, m)} \delta(e, e_i) \delta(f, f_j) \qquad (34)$$

# Training Model 2

$$\begin{aligned}
c_{\tilde{\theta}}(i \mid j, l, m; \mathbf{f}, \mathbf{e}) &= \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) c(i \mid j, l, m; \mathbf{a}, \mathbf{f}, \mathbf{e}) \\
&= \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \left( \prod_{k=1}^{m} p_{\tilde{\theta}}(a_k \mid k, \mathbf{f}, \mathbf{e}) \right) \delta(i, a_j) \\
&= \sum_{a_j=0}^{l} p_{\tilde{\theta}}(a_j \mid j, \mathbf{f}, \mathbf{e}) \delta(i, a_j) \qquad (35) \\
&= p_{\tilde{\theta}}(i \mid j, \mathbf{f}, \mathbf{e}) \\
&= \frac{p(f_j \mid e_i) p(i \mid j, l, m)}{\sum_{i'=0}^{l} p(f_j \mid e_{i'}) p(i' \mid j, l, m)} \qquad (36)
\end{aligned}$$





# Model 2: Training Algorithm

```
EM-MODEL2(F,m,E,l)
INIT-PARAMS(P,Q); // P[f,e]=p(f/e) Q[i,j,l,m]=p(i/j,l,m)
do
RESET-COUNTERS(p,q,ptot,qtot);
for s := 1 to S; // loop over training data
      do UPDATE-COUNTERS(F[s] LENGTH(F[s]),E[s],LENGTH(E[s]));
for m := 1 to M; //max source length
      do for l := 1 to L; // max target length
            do for j := 1 to m;
                  do for i := 0 to l;
                        do Q[i,j,l,m] := q[i,j,l,m]/qtot[j,l,m];
for f ∈ 𝓕;
      do for e ∈ 𝓔;
            do P[f,e] := p[f,e]/ptot[e];
until convergence
```

# Model 2: Training Algorithm

```
UPDATE-COUNTERS(F,m,E,l)
// Update counters p[], q[], ptot[], qtot[] using current parameters P[],Q[]
for j := 1 to m;
        do t := 0;
            for i := 0 to l;
                do f=F[j]; e=E[i];
                    t := t + P[f,e] * Q[i,j,l,m];
            for i := 0 to l;
                do f:=F[j]; e:=E[i];
                    c:= P[f,e] * Q[i,j,l,m] / t;
                    q[i,j,l,m] := q[i,j,l,m] + c;
                    qtot[j,l,m]=qtot[j,l,m] + c;
                    p[f,e] := p[f,e] + c;
                    ptot[e]:=ptot[e] + c;
```



FONDAZIONE BRUNO KESSLER


# Last Words About Model 2

- Re-estimation formulas of M2 require, for each translation sample, $\mathcal{O}(lm)$ computations, whereas the sum over alignments would need $\mathcal{O}((l+1)^m)$.
- Estimation of Model 1 corresponds to Model 2 with fixed alignment probs.
- **Computation of best alignment** for $(\mathbf{f}, \mathbf{e})$ with M2 is very fast, i.e.

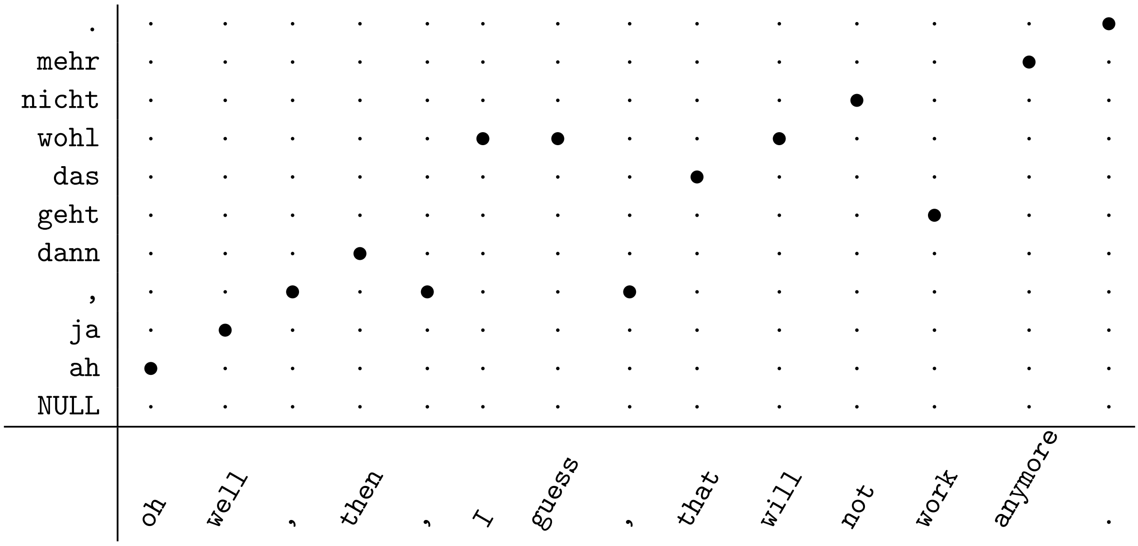
$$\mathbf{a}^* = \arg\max_{\mathbf{a}} \prod_{j=1}^{m} p(f_j \mid e_{a_j}) p(a_j \mid j, l, m) \quad (37)$$

$$a_j^* = \arg\max_{i} p(f_j \mid e_i) p(i \mid j, l, m) \quad (38)$$

- **Problems and limitations** of Model 1 and Model 2:
  – do not model the # of French words to be connected to each English word
  – the alignment probability scheme of Model 2 is complex and rigid

M. Federico, FBK-irst SMT - Part II Pisa, 7-19 May 2008

# Example of Best Alignment with Model 2

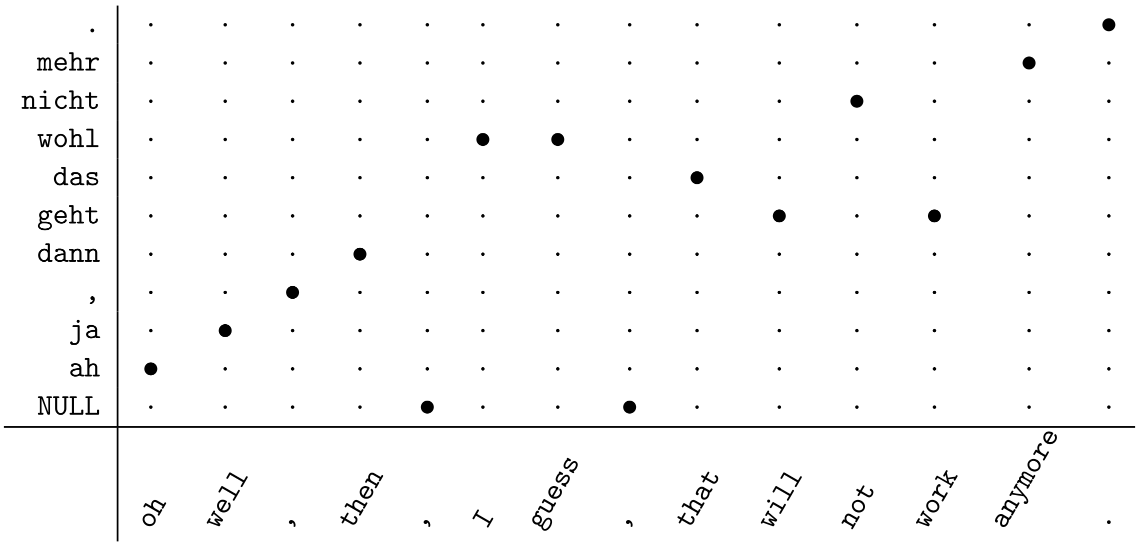



**Problems**: no coverage constraints for English words:
– words may be omitted or may be aligned to too many words





# Example: alignment with fertility models




**Fertility models** consider the number of words covered by each English word.

# Fertility Models

The number of French words covered by $e$ is a r.v. $\phi_e$: namely, the fertility of $e$.

- Models 1-2 do not explicitly model fertilities
- Models 3, 4, and 5 parameterize fertilities directly
- Fertility models imply a different **generative process** of $\mathbf{f}$ and $\mathbf{a}$ given $\mathbf{e}$:
  1. For $i = 1, \dots, l, 0$, choose a fertility value $\phi_i \geq 0$ for word $e_i$
  2. For $i = 1, \dots, l, 0$, choose a tablet $\tau_i$ of $\phi_i$ French words to translate $e_i$
  3. Choose a permutation $\pi$ over the tableau $\tau = (\tau_1, \dots, \tau_l, \tau_0)$ to generate $\mathbf{f}$
  4. **IF** any position was chosen more than once **THEN** return `FAILURE`
  5. **ELSE** return $(\mathbf{a}, \mathbf{f})$ corresponding to $(\tau, \pi)$.

**Notice:**

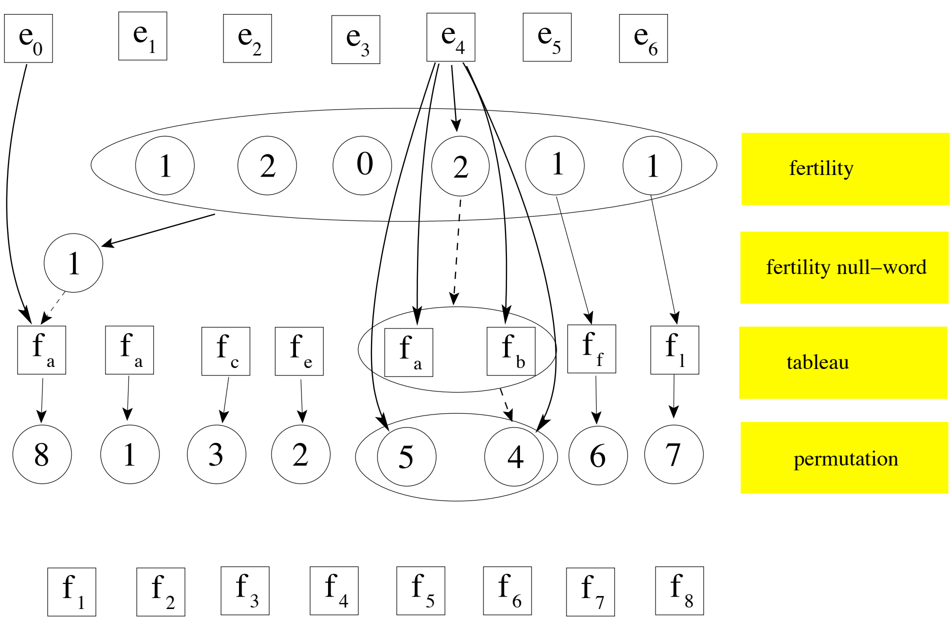
– for "correct" pairs $(\tau, \pi)$ there is a **many-to-one mapping** to $(\mathbf{f}, \mathbf{a})$.

– the notion of fertility is embedded into $\tau$ and $\pi$.





# IBM Model 3: Dependency Diagram



# IBM Model 3

The joint likelihood for a tableau $\tau$ and a permutation $\pi$ is:

$$\Pr(\tau, \pi \mid e_0^l) = \Pr(\phi_0^l \mid e_0^l) \cdot \Pr(\tau \mid \phi_0^l, e_0^l) \cdot \Pr(\pi \mid \tau, \phi_0^l, e_0^l)$$

We can distinguish three components, which are modeled as follows:

1. fertility generation (also for Models 4-5)

$$\Pr(\phi_0^l \mid e_0^l) = \prod_{i=1}^{l} P(\phi_i \mid e_i) \cdot p(\phi_0 \mid \sum_{i=1}^{l} \phi_i) \quad (39)$$

2. word generation (also for Models 4-5)

$$\Pr(\tau \mid \phi_0^l, e_0^l) = \prod_{i=0}^{l} \prod_{k=1}^{\phi_i} p(\tau_{ik} \mid e_i) \quad (40)$$





# IBM Model 3

The joint likelihood for a tableau $\tau$ and a permutation $\pi$ is:

$$\Pr(\tau, \pi \mid e_0^l) = \Pr(\phi_0^l \mid e_0^l) \cdot \Pr(\tau \mid \phi_0^l, e_0^l) \cdot \Pr(\pi \mid \tau, \phi_0^l, e_0^l)$$

We can distinguish three components, which are modeled as follows:

3. permutation generation (only for Model 3)

$$\Pr(\pi \mid \tau, \phi_0^l, e_0^l) = \frac{1}{\phi_0!} \cdot \prod_{i=1}^{l} \prod_{k=1}^{\phi_i} p(\pi_{ik} \mid i, l, m) \quad (41)$$

## Model 3: Fertility Generation

- Fertility of $e_i \quad i = 1, \ldots, l$ is drawn according to $p(\phi \mid e_i)$
- $e_i$ is called cept if it gets a fertility $\phi(e_i) > 0$
- Fertility of $e_0$ is generated according to:

$$\binom{\phi_1 + \ldots + \phi_l}{\phi_0} p_0^{\phi_1 + \ldots + \phi_l - \phi_0} p_1^{\phi_0} \qquad p_0 \geq 0, p_1 \geq 0, p_0 + p_1 = 1 \qquad (42)$$

**Interpretation:** among the $m - \phi_0$ French words generated by $\mathbf{e}$ there are $\phi_0$ words which need extra $\phi_0$ words to be connected to the empty word $e_0$. Implication: $\phi_0 \leq \frac{m}{2}$

- **Free parameters**: $p_1$ and $p_0$, $\{p(\phi \mid e) : e \in \mathcal{E}, \phi = 0, \ldots, \phi_{max}\}$.





## IBM Model 3: Word Generation

- Choose a French word $\tau_{ik} \quad i = 0, \ldots, l \quad k = 1, \ldots, \phi_i$ according to $p(\tau_{ik} \mid e)$.
- Only cepts and the empty word $e_0$ if $\phi_0 > 0$ do generate French words
- Free parameters: $\{p(f \mid e) : e \in \mathcal{E}, f \in \mathcal{F}\}$

# IBM Model 3: Permutation Generation

$$\Pr(\pi \mid \tau, \phi_0^l, e_0^l) = \prod_{i=1}^{l} \prod_{k=1}^{\phi_i} p(\pi_{ij} \mid i, l, m) \cdot \frac{1}{\phi_0!} \quad (43)$$

- Positions $\pi_{ik}$ are generated according to $p(\pi_{ik} \mid i, l, m)$

- Positions covered by the empty word are chosen at random assuming $\phi_0$ uncovered positions in the French string. Total probability is:

$$\prod_{k=1}^{\phi_0} p(\pi_{0\ k} \mid i, l, m) = (\phi_0)^{-1} \cdot (\phi_0 - 1)^{-1} \cdot \ldots \cdot 2^{-1} \cdot 1^{-1} = 1/\phi_0!$$

- Free parameters: those of $p(j \mid i, l, m)$, i.e. $(m_{max} \cdot l_{max})^2$.





# IBM Model 3: Alignment Model

- Generation of $(\tau, \pi)$ implies satisfaction of **exact coverage constraint**
- Many-to-one map from $(\tau, \pi)$ into $(\mathbf{f},\mathbf{a})$ is: $a_{\pi_{ik}} = i$ and $f_{\pi_{ik}} = \tau_{ik}$
- Simple re-arrangements within $(\tau, \pi)$ can result in the same pair $(\mathbf{f}, \mathbf{a})$, hence:

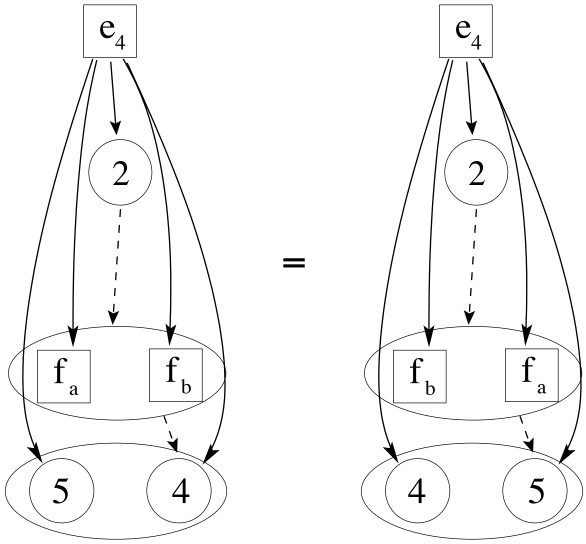
$$\Pr(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = \sum_{(\tau,\pi) \in (\mathbf{f},\mathbf{a})} Pr(\tau, \pi \mid \mathbf{e}) \quad (44)$$

- Each tablet $\tau_i$ and permutation $\pi_i$ can be re-arranged in $\phi_i!$ different ways to produce (with the same probability) the same pair $(\mathbf{f}, \mathbf{a})$. Hence:

$$\Pr(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = \binom{m - \phi_0}{\phi_0} p_0^{m-2\phi_0} p_1^{\phi_0} \cdot \prod_{i=1}^{l} p(\phi_i \mid e_i) \cdot \phi_i! \cdot \prod_{j=1}^{m} p(\mathbf{f}_j \mid e_{a_j}) \cdot \prod_{j:a_j>0} p(j \mid a_j, l, m)$$

we eliminate a factor $\phi_0!$ and introduce factors $\phi_i!$

# IBM Model 3: Tablet Permutation




The two processes generate the same alignment values for $a_4$ and $a_5$.





# IBM Model 3: Training

Similarly to Model 1-2, we get the following parameter re-estimation formulae:

$$c_{\tilde{\theta}}(f \mid e; \mathbf{f}, e) = \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{e}, \mathbf{f}) \sum_{j=1}^{m} \delta(f, f_j)\delta(e, e_{a_j}) \quad (45)$$

$$c_{\tilde{\theta}}(j \mid i; \mathbf{f}, e) = \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{e}, \mathbf{f})\delta(i, a_j) \quad (46)$$

$$c_{\tilde{\theta}}(\phi \mid e; \mathbf{f}, e) = \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{e}, \mathbf{f}) \sum_{i=1}^{l} \delta(\phi_i, \phi)\delta(e, e_i) \quad (47)$$

$$c_{\tilde{\theta}}(p_0; \mathbf{f}, e) = \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{e}, \mathbf{f})(m - 2\phi_0) \quad (48)$$

$$c_{\tilde{\theta}}(p_1; \mathbf{f}, e) = \sum_{\mathbf{a}} p_{\tilde{\theta}}(\mathbf{a} \mid \mathbf{e}, \mathbf{f})\phi_0 \quad (49)$$

The last two counts correspond to the number of French words with no extra French word, and to those with one extra English word.

# IBM Model 3: Training

- **Problem**: there is no trick to avoid explicit summation over all alignments
  – however: most of the alignments have very low probability
- **Trick**: summation over neighborhood of best (or Viterbi) alignment of M3.
- **Problem**: no efficient algorithm to compute the Viterbi alignment of M3
- **Trick**: do hill-climbing in space of possible alignments:
  – start from the Viterbi alignment of M2: $\mathbf{a}^* = V(\mathbf{f} \mid \mathbf{e}; M2)$
  – hill-climbing operator:

$$b(\mathbf{a}^*) = \arg \max_{\mathbf{a} \in \mathcal{N}(\mathbf{a}^*)} p(\mathbf{a} \mid \mathbf{e}, \mathbf{f}; M3) \qquad (50)$$

  – neighborhood $\mathcal{N}(\mathbf{a})$: alignments differing from $\mathbf{a}$ by one move or one swap
  – move operator: $m_{[j,i]}(\mathbf{a})$: changing $a_j := i$
  – swap operator: $s_{[j_1,j_2]}(\mathbf{a})$: exchanging $a_{j_1}$ with $a_{j_2}$





# IBM Model 3: Swaps

△= positions before swap ◇= positions after swap

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|
| $e_3$ | · | · | ● | · |
| $e_2$ | · | ● | · | ● |
| $e_1$ | ● | · | · | · |
| $e_0$ | · | · | · | · |

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|
| $e_3$ | · | · | ● | · |
| $e_2$ | ◇ | △ | · | ● |
| $e_1$ | △ | ◇ | · | · |
| $e_0$ | · | · | · | · |

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|
| $e_3$ | · | · | ● | · |
| $e_2$ | ◇ | ● | · | △ |
| $e_1$ | △ | · | · | ◇ |
| $e_0$ | · | · | · | · |

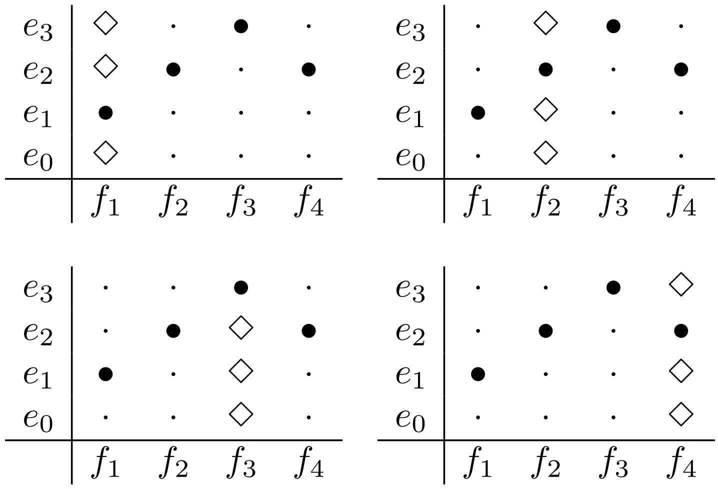
| | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|
| $e_3$ | ◇ | · | △ | · |
| $e_2$ | · | ● | · | ● |
| $e_1$ | △ | · | ◇ | · |
| $e_0$ | · | · | · | · |

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|
| $e_3$ | · | ◇ | △ | · |
| $e_2$ | · | △ | ◇ | ● |
| $e_1$ | ● | · | · | · |
| $e_0$ | · | · | · | · |

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|
| $e_3$ | · | · | △ | ◇ |
| $e_2$ | · | ● | ◇ | △ |
| $e_1$ | ● | · | · | · |
| $e_0$ | · | · | · | · |

Total number of swaps: $\leq m \cdot (m-1)/2$
Maximum is reached when all source pairs map to different positions!

# IBM Model 3: Moves




- Number of moves: $m \times l$.

**Total number of elements in a neighborhood** (must include **a** itself) is:
$|\mathcal{N}(\mathbf{a})| \leq 1 + m \cdot l + m \cdot (m-1)/2$.





# IBM Model 3: Training with constraints

- **Pegged (or constrained) Viterbi alignment**:

$$V_{i \leftarrow j}(\mathbf{f} \mid \mathbf{e}; M2) = \arg \underset{\mathbf{a}: a_j = i}{max} \Pr(\mathbf{a} \mid \mathbf{e}, \mathbf{f}; M2) \quad (51)$$

- **Pegged hill climbing operator for Model 3**:

$$b_{i \leftarrow j}(\mathbf{a}) = \arg \underset{\mathbf{a}' \in \mathcal{N}(\mathbf{a}) \cap a'_j = i}{max} \Pr(\mathbf{a}' \mid \mathbf{e}, \mathbf{f}; M3) \quad (52)$$

- Repeated application of $b(\cdot)$ and $b_{i \leftarrow j}(\cdot)$ always converges
- IBM recipe is to reestimate parameters with the following set of alignments $\mathcal{S}$:

$$\mathcal{S} = \mathcal{N}(b^{\infty}(V(\mathbf{f} \mid \mathbf{e}; M2))) \bigcup \cup_{ij} \mathcal{N}(b^{\infty}_{i \leftarrow j}(V_{i \leftarrow j}(\mathbf{f} \mid \mathbf{e}; M2))) \quad (53)$$

Several variations have been proposed in subsequent papers.

# IBM Model 3: Training (w/o pegging)

1. Calculate Viterbi alignment of Model 2:
   $a_0 = V(\mathbf{f} \mid \mathbf{e}; M2); i = 0$
2. Hill-climbing
   do; $i = 1 + 1$; $\mathbf{a}_i = b(\mathbf{a}_{i-1})$;while $\mathbf{a}_i \neq \mathbf{a}_{i-1}$;
3. Update counters
   for each $\mathbf{a}$ in $\mathcal{N}(\mathbf{a}_i)$; do $t+ = p_{\tilde{\theta}}(\mathbf{a}, \mathbf{f} \mid \mathbf{e}; M3)$;
   for each $\mathbf{a}$ in $\mathcal{N}(\mathbf{a}_i)$; do
      $p = p_{\tilde{\theta}}(\mathbf{a}, \mathbf{f} \mid \mathbf{e}; M3)/t$;
      for $j = 1$ to $m$; do $c_{\tilde{\theta}}(j \mid a_j; \mathbf{f}, \mathbf{e})+ = p$;
      for $i = 1$ to $l$; do $c_{\tilde{\theta}}(\phi_i \mid e_i; \mathbf{f}, \mathbf{e})+ = p$;
      for $j = 1$ to $m$; do $c_{\tilde{\theta}}(f_j \mid e_{a_j}; \mathbf{f}, \mathbf{e})+ = p$;
      $c_{\tilde{\theta}}(p_0; \mathbf{f}, \mathbf{e})+ = p \cdot (m - 2\phi_0)$
      $c_{\tilde{\theta}}(p_1; \mathbf{f}, \mathbf{e})+ = p \cdot \phi_0$
4. ...





# IBM Model 3: Training algorithm

- Computing $p_{\tilde{\theta}}(\mathbf{a}, \mathbf{f} \mid \mathbf{e}; 3)$ requires $O(m + l)$ multiplications
- After a move/swap, savings can be obtained by incremental computation
- Let us assume a move of position $j$ from $i$ to $i'$, with $i \neq i' \neq 0$:

$$\frac{\Pr(m_{j,i'}(\mathbf{a}), \mathbf{f} \mid \mathbf{e})}{\Pr(\mathbf{a}, \mathbf{f} \mid \mathbf{e})} = \frac{\phi_{i'} + 1}{\phi_i} \cdot \frac{p(\phi_{i'} + 1 \mid e_{i'})}{p(\phi_{i'} \mid e_{i'})} \cdot \frac{p(\phi_i - 1 \mid e_i)}{p(\phi_i \mid e_i)} \cdot \frac{p(f_j \mid e_{i'})}{p(f_j \mid e_i)} \cdot \frac{p(j \mid i', m, l)}{p(j \mid i, m, l)} \quad (54)$$

- Let us assume a swap of position $j_1$ and $j_2$, with $0 < i_1 = a_{j_1} \neq a_{j_2} = i_2 > 0$:

$$\frac{\Pr(s_{j_1,j_2}(\mathbf{a}), \mathbf{f} \mid \mathbf{e})}{\Pr(\mathbf{a}, \mathbf{f} \mid \mathbf{e})} = \frac{p(f_{j_1} \mid e_{i_2})}{p(f_{j_1} \mid e_{i_1})} \cdot \frac{p(f_{j_2} \mid e_{i_1})}{p(f_{j_2} \mid e_{i_2})} \cdot \frac{p(j_1 \mid i_2, m, l)}{p(j_1 \mid i_1, m, l)} \cdot \frac{p(j_2 \mid i_1, m, l)}{p(j_2 \mid i_2, m, l)} \quad (55)$$

- The number of multiplications is now 10 and 8, respectively. Similar relations hold for the conditions we have left out.

# IBM Model 4

- Model 3 assumes independence between alignment positions
- Model 4 changes the permutation model as follows

$$\Pr(\pi \mid \tau, \phi_0^l, e_0^l) = \frac{1}{\phi_0!} \cdot \prod_{i=1}^{l} \prod_{k=1}^{\phi_i} p(\pi_{ik} \mid i, l, m) \quad \text{(Model 3)} \tag{56}$$

$$\Pr(\pi \mid \tau, \phi_0^l, e_0^l) = \frac{1}{\phi_0!} \cdot \prod_{i=1}^{l} \prod_{k=1}^{\phi_i} p(\pi_{ik} \mid \pi_1^{i-1}, \tau_i, \mathbf{e}) \quad \text{(Model 4)} \tag{57}$$

- In the generation of the permutation we distinguish the cases:
  - $k = 1$: choice of first position in the tablet
  - $k > 1$: choice of subsequent positions in the tablet
- The **probability depends on the distance** between covered positions
- Such a model is also called distortion model



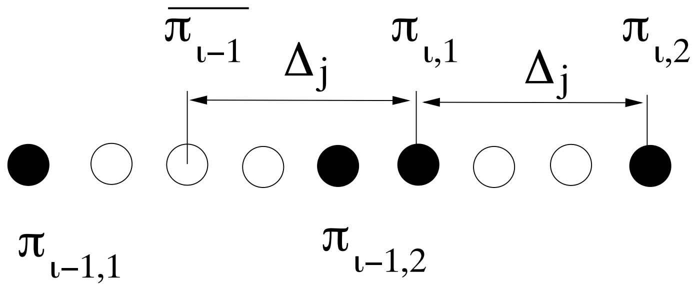


# IBM Model 4: permutation model

We indicate by circles positions in the source string:

- filled circle= covered position
- empty circle = not yet covered position

If we are covering the first position of tablet $\pi_i$, the distortion probability depends on the "distance" from the center $\bar{\pi}_{i-1}$ of the last permutation $\pi_{i-1}$, otherwise it depends on the "distance" from the last chosen position.



# IBM Model 4: permutation model

- When generating the first position for word $e$ we consider the **offset** from the center of positions corresponding to the last English cept.
- When generating subsequent positions for word $e$ we consider the **offset** from the last position chosen for $e$.

$$p(\pi_{ik} \mid \pi_1^{i-1}, \tau_i, \mathbf{e}) = \begin{cases} p_{=1}(\pi_{i1} - \bar{\pi}_{\rho(i)} \mid \mathcal{A}(e_{\rho(i)}), \mathcal{B}(\tau_{i1})) & \text{if } k = 1 \\ p_{>1}(\pi_{ik} - \pi_{ik-1} \mid \mathcal{B}(\tau_{ik})) & \text{if } k > 1 \end{cases} \quad (58)$$

where:

$$\rho(i) = \max_{i'<i} \{i' : \phi_i > 0\} \quad \text{and} \quad \bar{\pi}_i = \left\lceil \frac{1}{\phi_i} \sum_{k=1}^{\phi_i} \pi_{ik} \right\rceil \quad (59)$$

$$p_{>1}(x \mid \mathcal{B}(f)) = 0 \text{ if } x < 0 \quad \text{(monotonic permutation)} \quad (60)$$

$\mathcal{A}(e)$ and $\mathcal{B}(f)$ are word classes for English and French words (by some clustering)





# IBM Model 4 Training

- Due to the monotonic permutation constraint we have a **one-to-one mapping** between consistent $(\tau, \pi)$ and $(\mathbf{a}, \mathbf{f})$
- Training of Model 4 has the same problems of Model 3.
- Incremental computation of $\Pr(\mathbf{a}, \mathbf{f} \mid \mathbf{e})$ is more complicated: moving French words from one cept to another might change the cept centers!
- We use a **different hill-climbing operator** starting from $V(\mathbf{f} \mid \mathbf{e}; M2)$
  – rank neighbors in $\mathcal{N}(\mathbf{a})$ according to $\Pr(\mathbf{a} \mid \mathbf{f}, \mathbf{e}; M3)$
  – $\tilde{b}(\mathbf{a}) \equiv$ first neighbor of $\mathbf{a}$ s.t. $\Pr(\tilde{b}(\mathbf{a}) \mid \mathbf{f}, \mathbf{e}; M4) \geq \Pr(\mathbf{a} \mid \mathbf{f}, \mathbf{e}; M4)$
  – idea: rank with M3 and rescore with M4
- We define similarly an operator $\tilde{b}_{i \leftarrow j}(\mathbf{a})$
- Hence, we define $\mathcal{S}$ for Model 4 by:

$$\mathcal{S} = \mathcal{N}(\tilde{b}^{\infty}(V(\mathbf{f} \mid \mathbf{e}; M2))) \bigcup \cup_{ij} \mathcal{N}(\tilde{b}^{\infty}_{i \leftarrow j}(V_{i \leftarrow j}(\mathbf{f} \mid \mathbf{e}; M2))) \quad (61)$$

## Deficiency

- Model 3 and Model 4 are so-called deficient, i.e.:

$$\sum_{\mathbf{f}} \Pr(\mathbf{f} \mid \mathbf{e}) < 1 \;\; \text{and} \;\; \Pr(\texttt{failure} \mid \mathbf{e}) > 0 \tag{62}$$

- Model 3: it checks that all positions in the source are covered exactly once
- Model 4: idem + check source positions are within the limits
- Deficiency poses no serious problem!
- Model 5 eliminates deficiency by keeping track of free positions.





## IBM Model 5

Model 5 generates positions in a fixed order **top-down left-to-right**:

$$\Pr(\pi \mid \tau, \phi, \mathbf{e}) = \frac{1}{\phi_0!} \prod_{i=1}^{l} \prod_{k=1}^{\phi_i} \Pr(\pi_{ik} \mid \phi_0^l, \tau_0^l, \pi_1^{i-1}, \pi_{i1}, \ldots, \pi_{ik-1}) \tag{63}$$

- The model keeps track of uncovered positions:
  $\epsilon_{ik}(j)$ is 1 if position $j$ is vacant just before placing $\tau_{ik}$, and 0 otherwise

$$p(\pi_{ik} = j \mid \ldots) = \epsilon_{ik}(j) \begin{cases} p_{=1}(.. \mid ..) & \text{if } k = 1 \\ p_{>1}(.. \mid ..) & \text{otherwise} \end{cases}$$

- The distortion probability is positive only if $j$ is vacant!
- **Remark**: there are some inconsistencies in the IBM paper (text vs. appendix) about the model for case $k = 1$: i.e. only the model in the text is not deficient.

# IBM Model 5: A Closer Look ($k = 1$)

Let

- $v_{ik}(j) = \sum_{j' \leq j} \epsilon_{ik}(j')$ be the number of vacancies up to position $j$
- $c_{\rho_i}$ be the center of last cept generated before $i$

The probability for placing the first word of tablet $\tau_i$ in vacant position $j$:

$$p_{=1}\left(v_{i1}(j) \mid \mathcal{B}(\tau_{i1}), v_{i1}(c_{\rho_i}), v_{i1}(m) - \phi_i + k\right) \tag{64}$$

- $v_{i1}(j)$ tells the chosen vacancy from left-to-right
- $\mathcal{B}(f)$ is a lexical class defined for French word $f$
- $v_{i1}(m) - \phi_i + k$ is the number of available vacancies to place $\phi_i$ words
- This definition permits to assign zero probability to forbidden positions j i.e. positions for which there are not enough vacancies on the right side
- Dependency is given on the centre $c_{\rho_i}$ in terms of vacancies (not clear why)





# IBM Model 5: A Closer Look ($k > 1$)

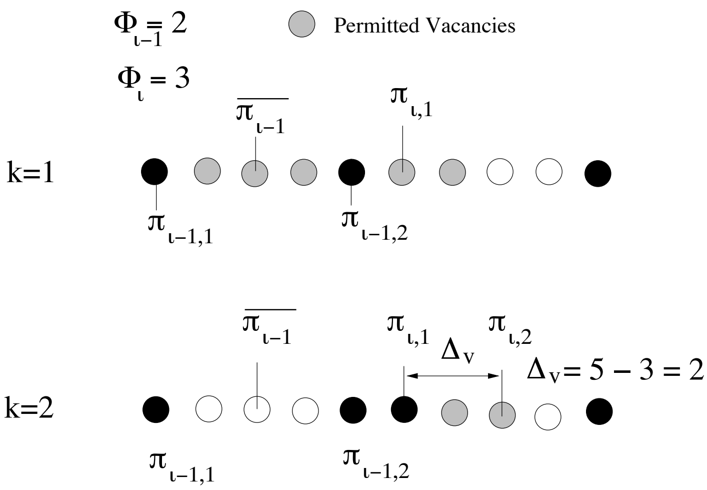
The probability of choosing vacancies to the right is:

$$p_{>1}\left(v_{ik}(j) - v_{ik}(\pi_{ik-1}) \mid \mathcal{B}(\tau_{ik}), v_{ik}(m) - v_{ik}(\pi_{ik-1}) - \phi_i + k\right) \tag{65}$$

- $v_{ik}(j) - v_{ik}(\pi_{ik-1})$: is the gap in terms of vacancies from previous placement
- $v_{ik}(m) - v_{ik}(\pi_{ik-1}) - \phi_i + k$: is maximum allowed vacancy gap for this position
- This definition permits to assign zero probability to forbidden placements $j$ i.e. positions for which there are not enough vacancies left on the right side
- Dependencies are different in type from those of IBM Model 4, but number of parameters is comparable.

IBM Model 5 has proven to provide more accurate alignments but its impact on translation accuracy is not significant.

# Model 5: permutation model







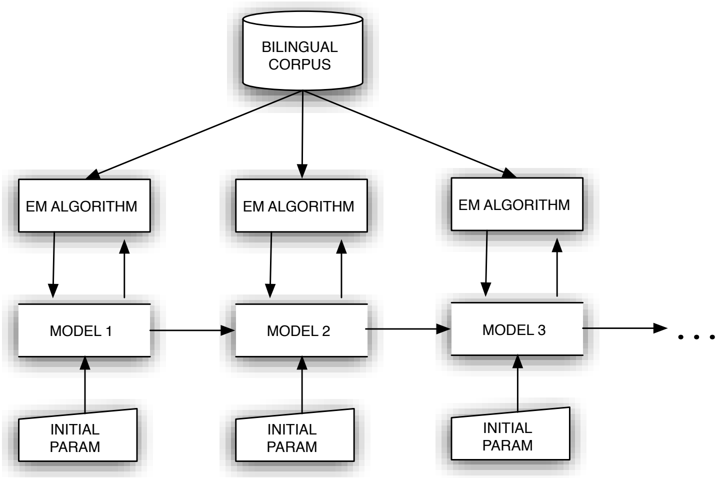

# Summary on Training of Alignment Models

- Training of an alignment model means finding good values for its basic probabilities or parameters, which we indicate by $\theta$.
- In general, given a sample of translations $\{(\mathbf{f}_s, \mathbf{e}_s) : s = 1, \ldots, S\}$, we would like to find parameter values $\theta$ which maximize the log-likelihood function:

$$\psi(p_\theta) = S^{-1} \sum_{s=1}^{S} \log p_\theta(\mathbf{f}_s \mid \mathbf{e}_s) \quad (66)$$

- Maximum likelihood (ML) estimates could be easily computed if the parallel corpus would include word alignments: just use relative frequencies!
- If word alignments are not available, ML estimates can be computed with the so called EM (Expectation Maximization) algorithm.
- For some models approximations of the EM algorithm are considered to make computations feasible.

# Incremental Training Procedure




use previous model to initialize some of the parameters of next model!





# HMM Alignment Model

**Another alignment model** which follows from the general alignment model:

$$\Pr(f_1^m, a_1^m \mid e_1^l) = \Pr(m \mid e_1^l) \prod_{j=1}^{m} \Pr(a_j \mid f_1^{j-1}, a_1^{j-1}, m, e_1^l) \cdot \Pr(f_j \mid f_1^{j-1}, a_1^j, m, e_1^l)$$

Let us define the following parameters:

$$\begin{array}{rll} \Pr(m \mid e_1^l) & = p(m \mid l) & \text{string length probabilities} \\ \Pr(a_j \mid f_1^{j-1}, a_1^{j-1}, m, e_1^l) & = p(a_j \mid a_{j-1}, l) & \text{alignment probabilities} \\ \Pr(f_j \mid f_1^{j-1}, a_1^j, m, e_1^l) & = p(f_j \mid e_{a_j}) & \text{translation probabilities} \end{array}$$

Hence, we get the following translation model:

$$\Pr(f_1^m \mid e_1^l) = \sum_{a_1^m} \Pr(f_1^m, a_1^m \mid e_1^l) = p(m \mid l) \cdot \sum_{a_1^m} \prod_{j=1}^{m} p(a_j \mid a_{j-1}, l) \cdot p(f_j \mid e_{a_j})$$

# HMM: Alignment Probabilities

The alignment probability is modeled such that:

- if $a_j \neq 0$ then $p(a_j \mid ...)$ depends on the most recent non empty alignment.
- if $a_j = 0$ then the probability $P(a_j \mid ...)$ is constant

**There is a trick**: alignment range is extended to $\{1, \ldots, l, l+1, \ldots, 2l\}$

- positions $> l$ are used to remember the most recent non empty position

$$p(a_j \mid a_{j-1}) = \begin{cases} p_0 & \text{if } a_j = a_{j-1} + l \text{ (to null)} \\ p_0 & \text{if } a_j > l \text{ and } a_j = a_{j-1} \text{ (cont. null)} \\ (1-p_0)p'(a_j \mid a_{j-1} - l) & \text{if } a_j \leq l \text{ and } a_{j-1} > l \\ (1-p_0)p'(a_j \mid a_{j-1}) & \text{if } a_j \leq l \text{ and } a_{j-1} \leq l \\ 0 & \text{otherwise} \end{cases}$$





# HMM: Alignment Probabilities

**The alignment probability depends on relative distances between positions**, not on absolute positions!

Transition model is so called homogeneous, i.e.

$$p'(i \mid i', l) = \frac{p(i - i')}{\sum_{i''=1}^{l} p(i'' - i')} \quad (67)$$

- We end up with a distortion model defined by a table of $max_l$ entries.
- HMM alignment model can be trained with an approximate EM algorithm.
- In the incremental training procedure, HMM are put between IBM 2 and IBM3

# How do we use alignments for?

Given a parallel corpus we can compute Viterbi alignments with some models to:

- discover interesting lexical relationships
- generate a probabilistic translation lexicon
- to extract phrase-pairs

Alignments have limitations in terms of allowed word mappings, it is widely known that better alignments can be obtained by:

- estimating alignments from source to target and viceversa
- computing a suitable combination of the two alignments

In the following, we will see different ways to combine alignments