

A stylized, monochromatic yellow illustration of a city skyline, featuring various skyscrapers and buildings of different heights and shapes. The background is a solid light yellow color.

statistical machine translation

PART 2: IBM MODEL

CSC401/2511 – Natural Language Computing – Spring 2019

Lecture 6 Frank Rudzicz and Chloé Pou-Prom

University of Toronto

Statistical Machine Translation

- Challenges to statistical machine translation
- Sentence alignment
- IBM model
- Phrase-based translation
- Decoding
- Evaluation

How to use the noisy channel

- How does this work?

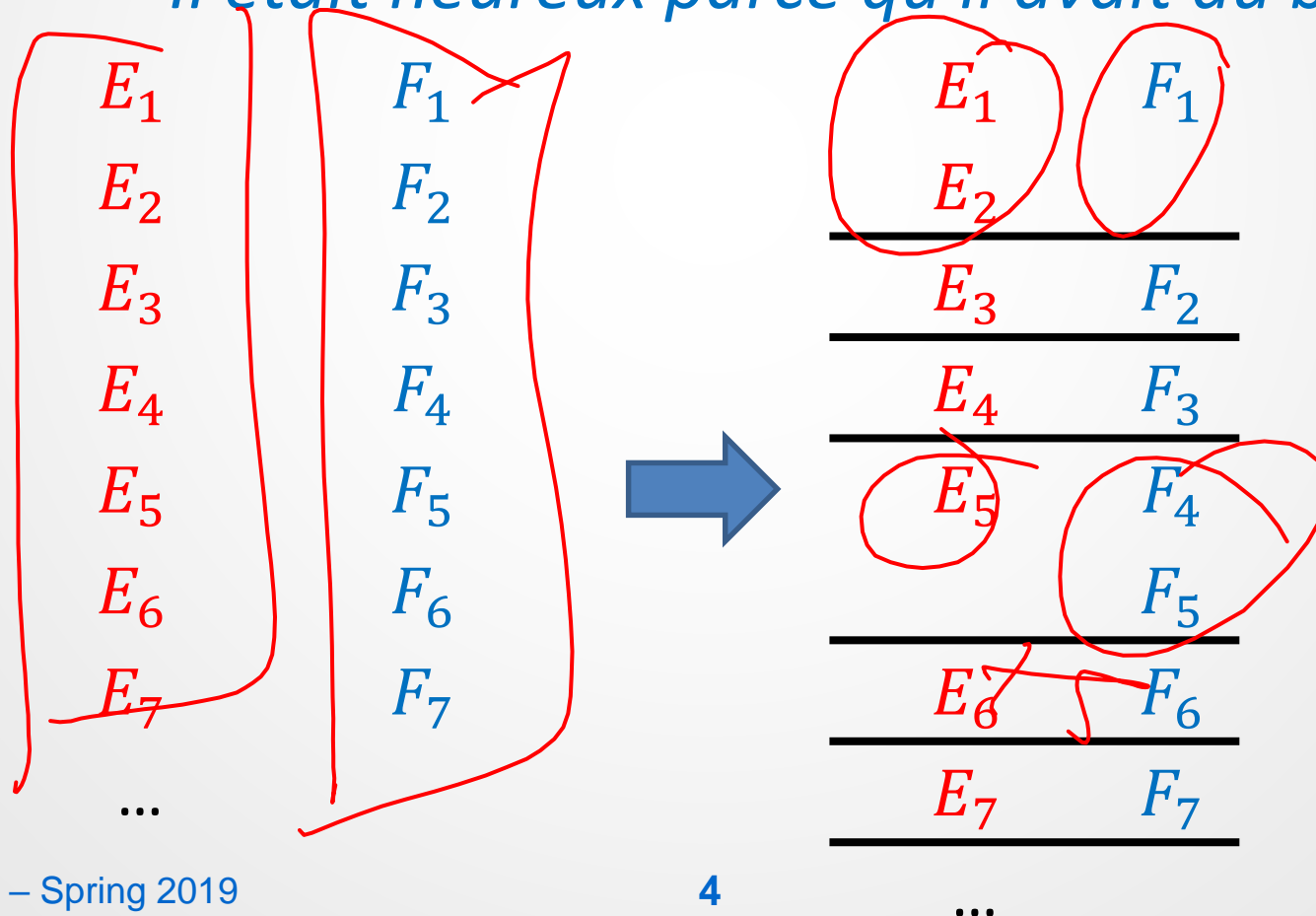
$$E^* = \operatorname{argmax}_E P(F|E)P(E)$$

- $P(E)$ is a **language model** (e.g., N -gram) and encodes knowledge of word order.
- $P(F|E)$ is a **word-level translation model** that encodes only knowledge on an *unordered* word-by-word basis.
- **Combining** these models can give us **naturalness** and **fidelity**, respectively.

Sentence alignment

- Sentences can also be **unaligned** across translations.

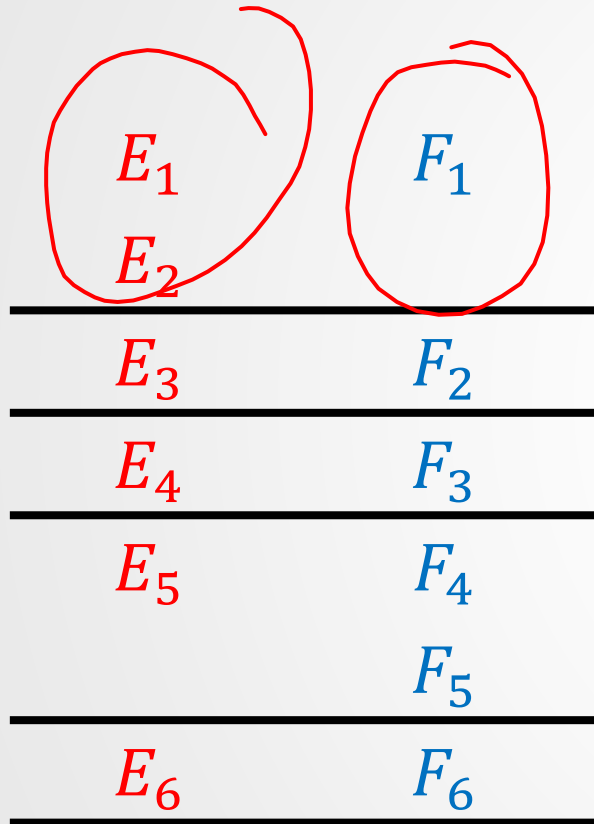
• E.g., *He was happy.*_{E₁} *He had bacon.*_{E₂} →
*Il était heureux parce qu'il avait du bacon.*_{F₁}



Sentence alignment

- We often need to align **sentences** before we can align **words**.
- We'll look at two broad classes of methods:
 1. Methods that only look at **sentence length**,
 2. Methods based on **lexical matches**, or “cognates”.

1. Sentence alignment by length



It's a bit more complicated – see paper on course webpage

We can associate costs with different **types** of alignments.

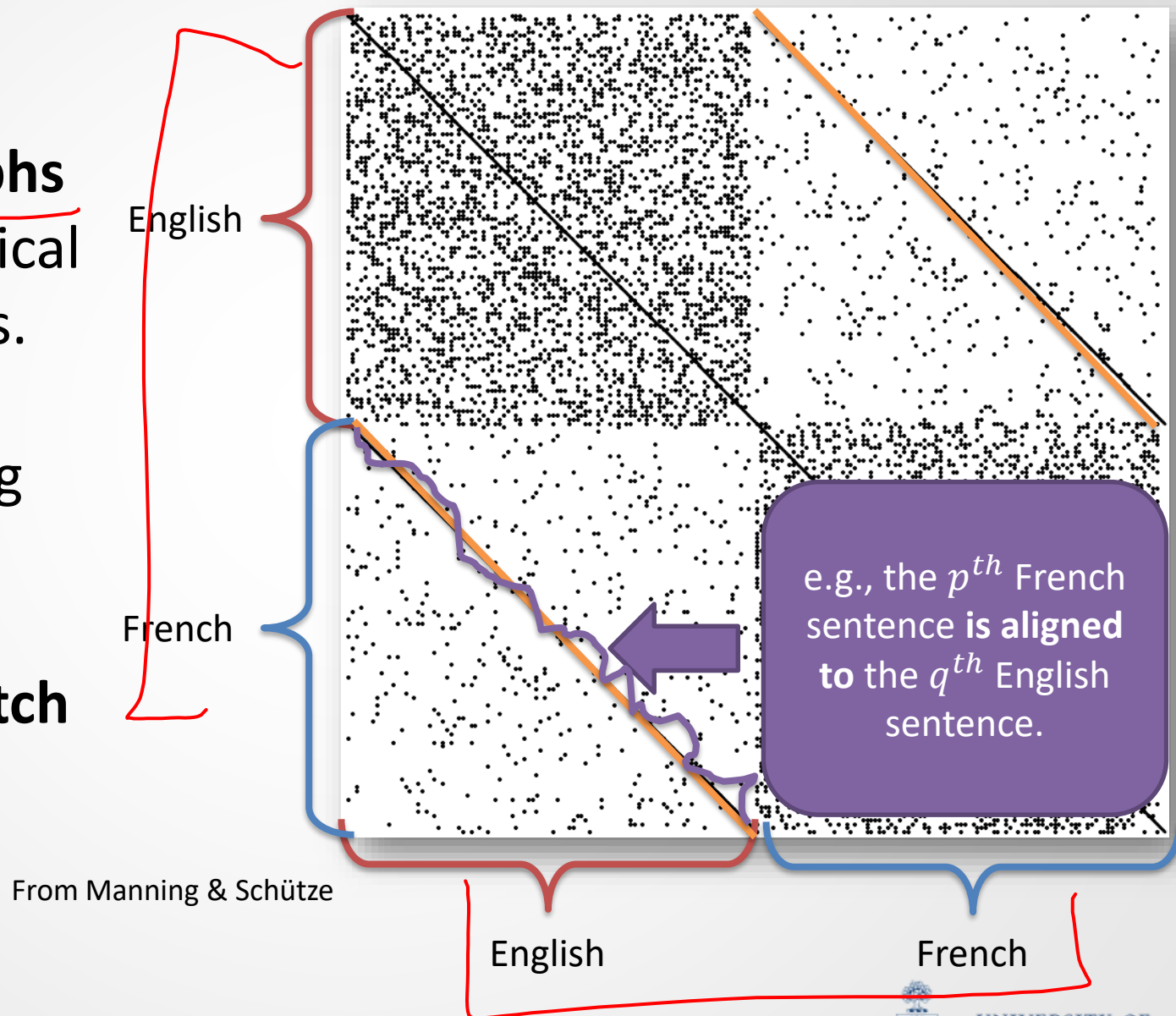
$C_{i,j}$ is the prior cost of aligning i sentences to j sentences.

$$\begin{aligned} \text{Cost} = & \underbrace{\text{Cost}(\mathcal{L}_{E_1} + \mathcal{L}_{E_2}, \mathcal{L}_{F_1})}_{\text{aligning 2 sentences to 1}} + \underbrace{C_{2,1}}_{\text{prior cost}} + \\ & \text{Cost}(\mathcal{L}_{E_3}, \mathcal{L}_{F_2}) + C_{1,1} + \\ & \text{Cost}(\mathcal{L}_{E_4}, \mathcal{L}_{F_3}) + C_{1,1} + \\ & \text{Cost}(\mathcal{L}_{E_5}, \mathcal{L}_{F_4} + \mathcal{L}_{F_5}) + C_{1,2} + \\ & \text{Cost}(\mathcal{L}_{E_6}, \mathcal{L}_{F_6}) + C_{1,1} \end{aligned}$$

Find distribution of sentence breaks with minimum cost using **dynamic programming**

2a. Church's method

- Church (1993) tracks all **4-graphs** which are identical across two texts.
- Each point along **this path** is considered to represent a **match** between languages.



2b. Melamed's method

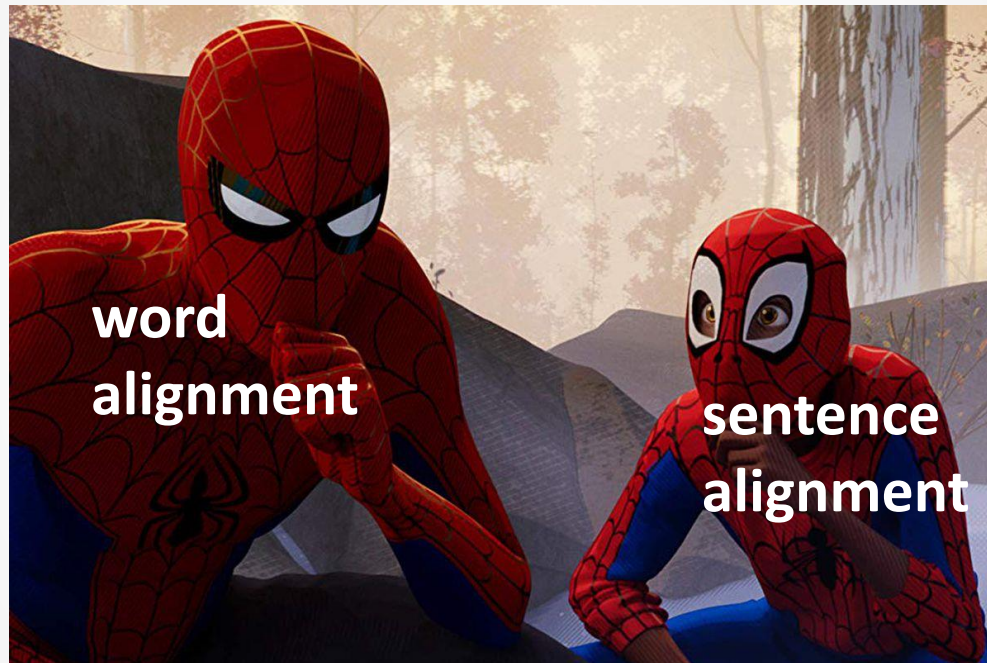
- $LCS(A, B)$ is the longest common subsequence of characters (with gaps allowed) in words A and B .
- Melamed (1993) measures similarity of words A and B

$$LCSR(A, B) = \frac{\text{length}(LCS(A, B))}{\max(\text{length}(A), \text{length}(B))}$$

- e.g.,

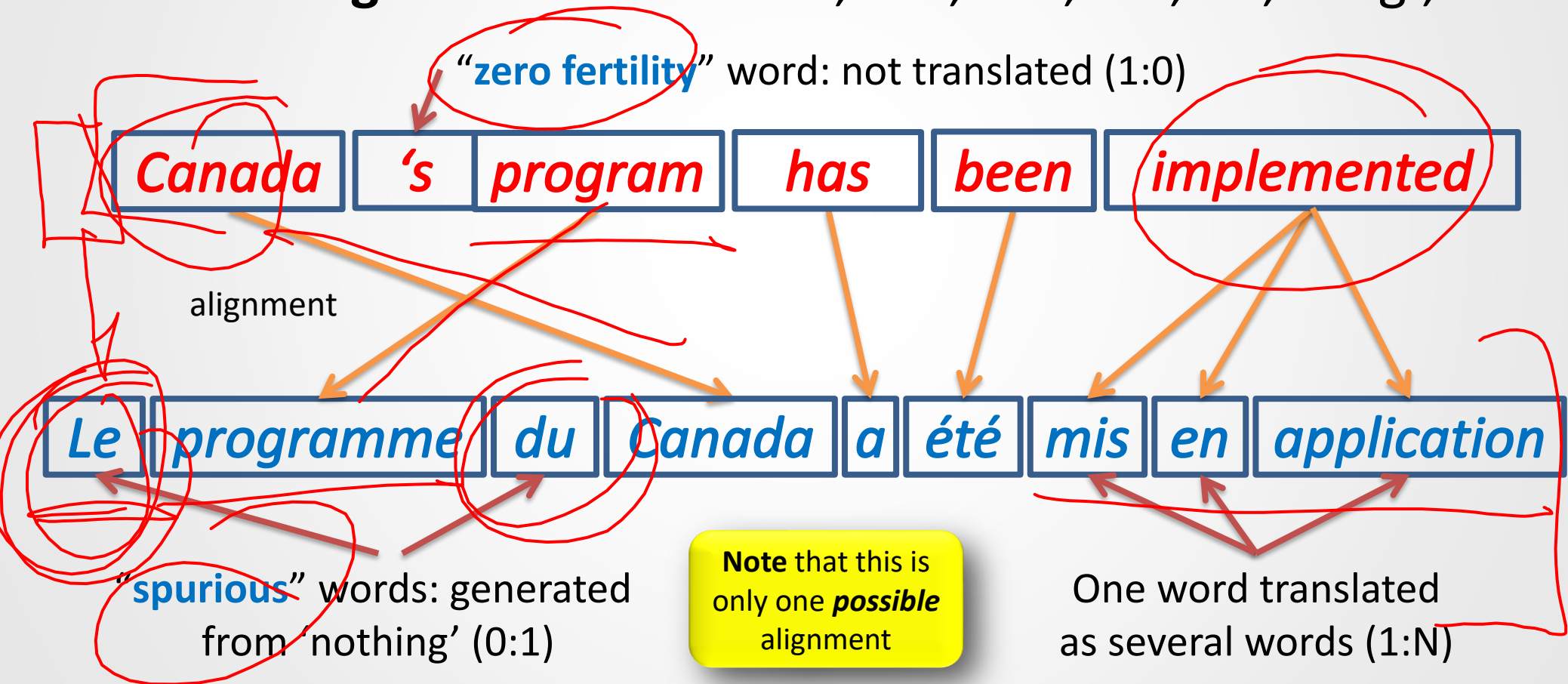
$$LCSR(\text{government}, \text{gouvernement}) = \frac{10}{12}$$

'LCS Ratio'



Word alignment

- **Word alignments** can be 1:1, N:1, 1:N, 0:1, 1:0,... E.g.,



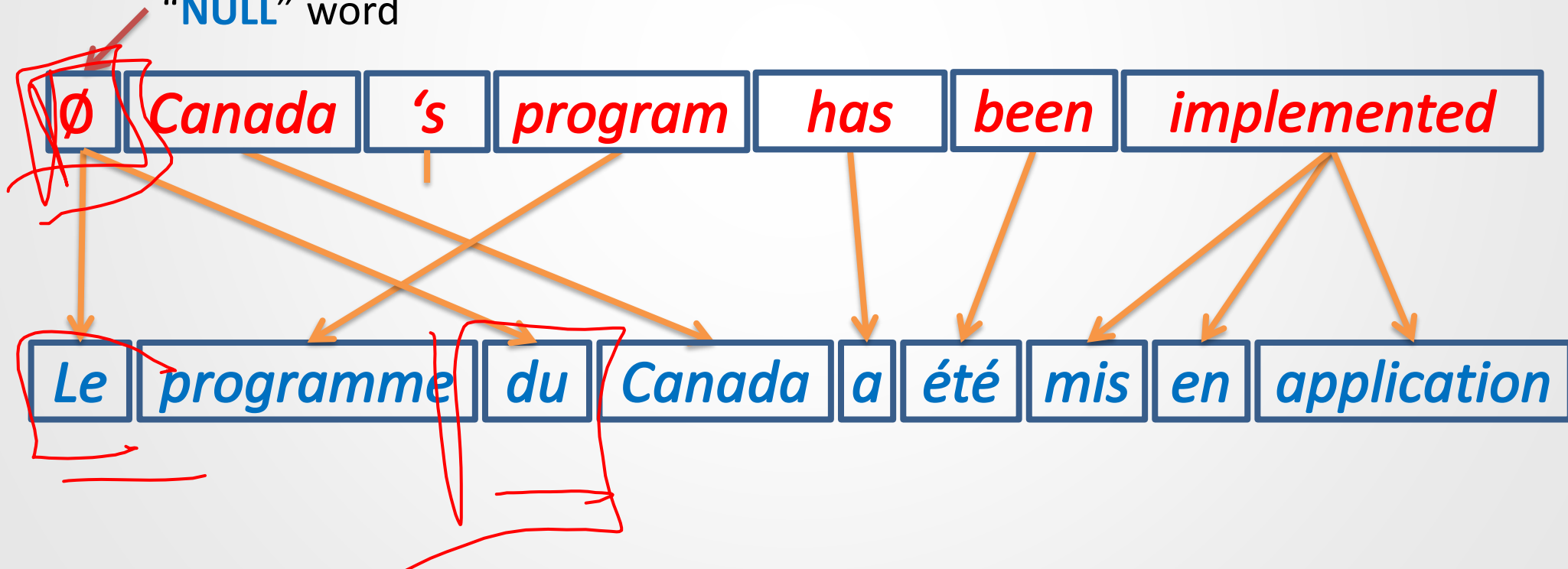
$p(f|e)$

IBM Model 1

IBM Model 1: the NULL word

- The **NULL** word is an imaginary word that we need to account for the production of spurious words.

“NULL” word



IBM Model 1: some definitions

- English sentence E has L_E words, $e_1 \dots e_{L_E}$,
plus NULL word, e_0 .
- French sentence F has L_F words, $f_1 \dots f_{L_F}$.

| e_0 | e_1 | e_2 | e_3 | e_4 | e_5 | e_6 |
|-------------|--------|-------|---------|-------|-------|-------------|
| \emptyset | Canada | 's | program | has | been | implemented |

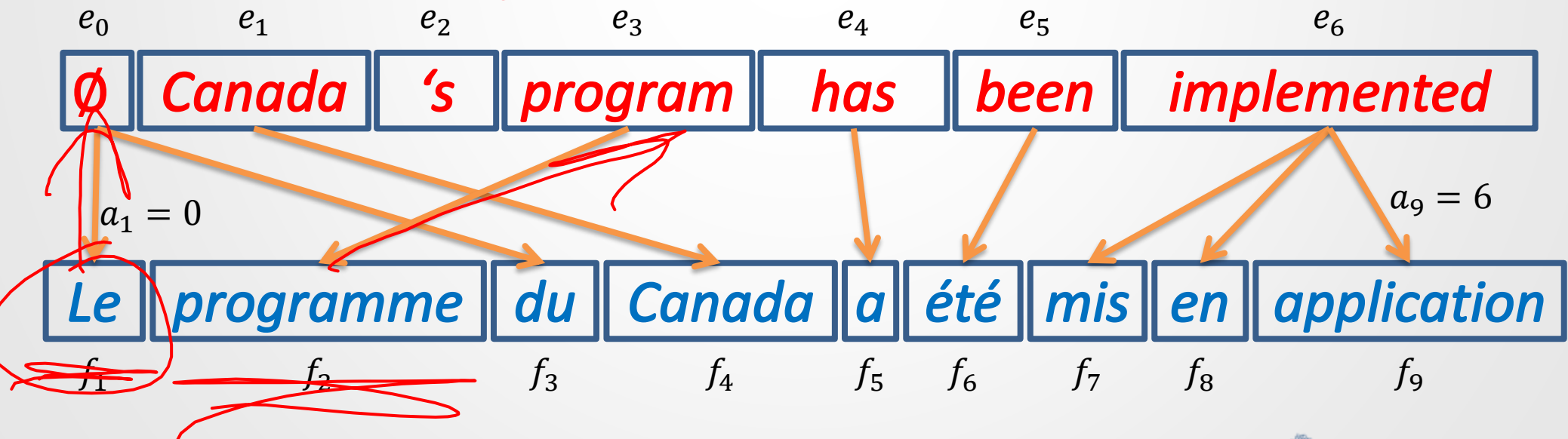
| | | | | | | | | |
|-------|-----------|-------|--------|-------|-------|-------|-------|-------------|
| Le | programme | du | Canada | a | été | mis | en | application |
| f_1 | f_2 | f_3 | f_4 | f_5 | f_6 | f_7 | f_8 | f_9 |

IBM Model 1: alignments

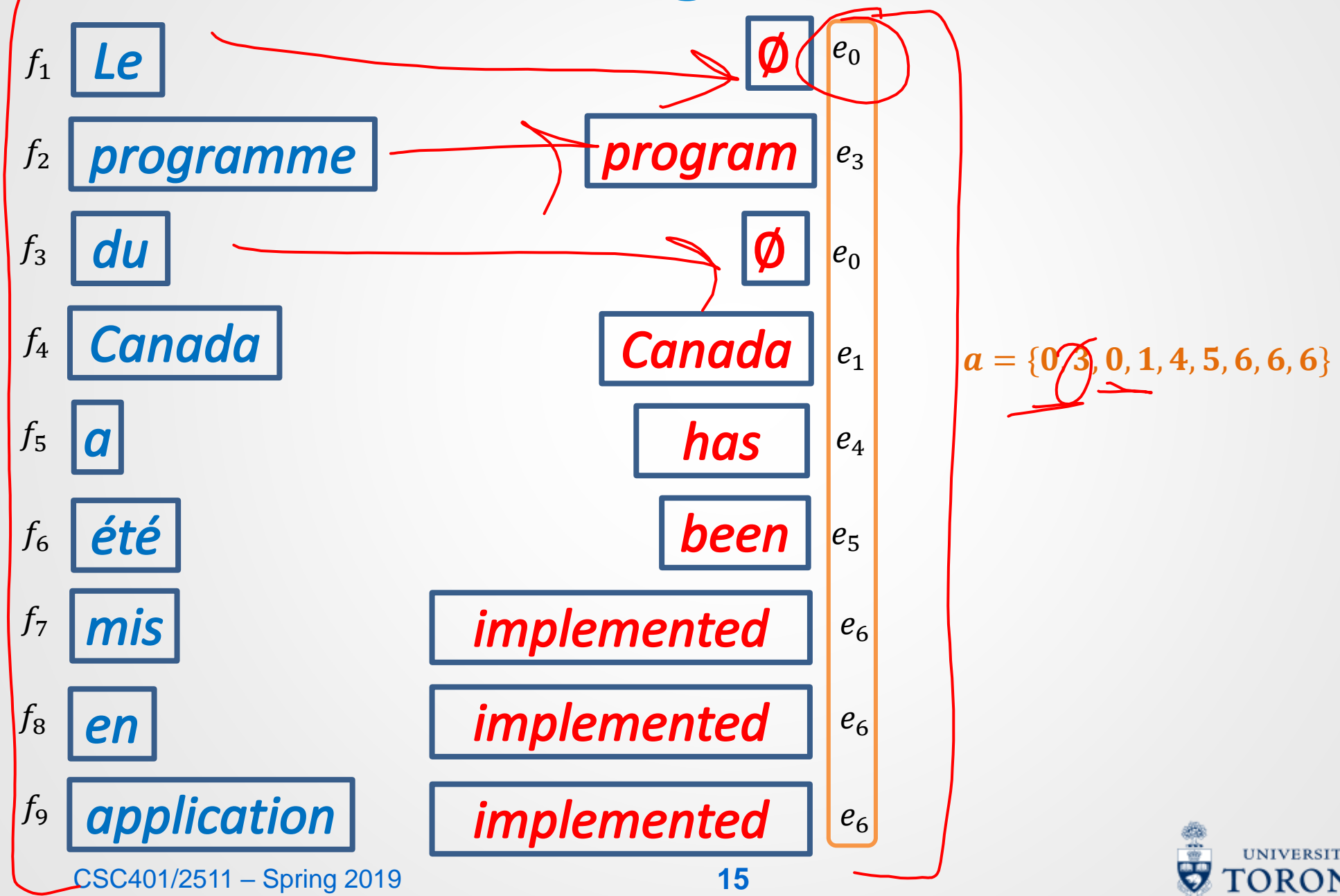
- An **alignment**, a , identifies the English word that ‘produced’ the given French word at each index.

- $a = \{a_1, \dots, a_{L_F}\}$ where $a_j \in \{0, \dots, L_E\}$

- E.g., $a = \{0, 3, 0, 1, 4, 5, 6, 6, 6\}$



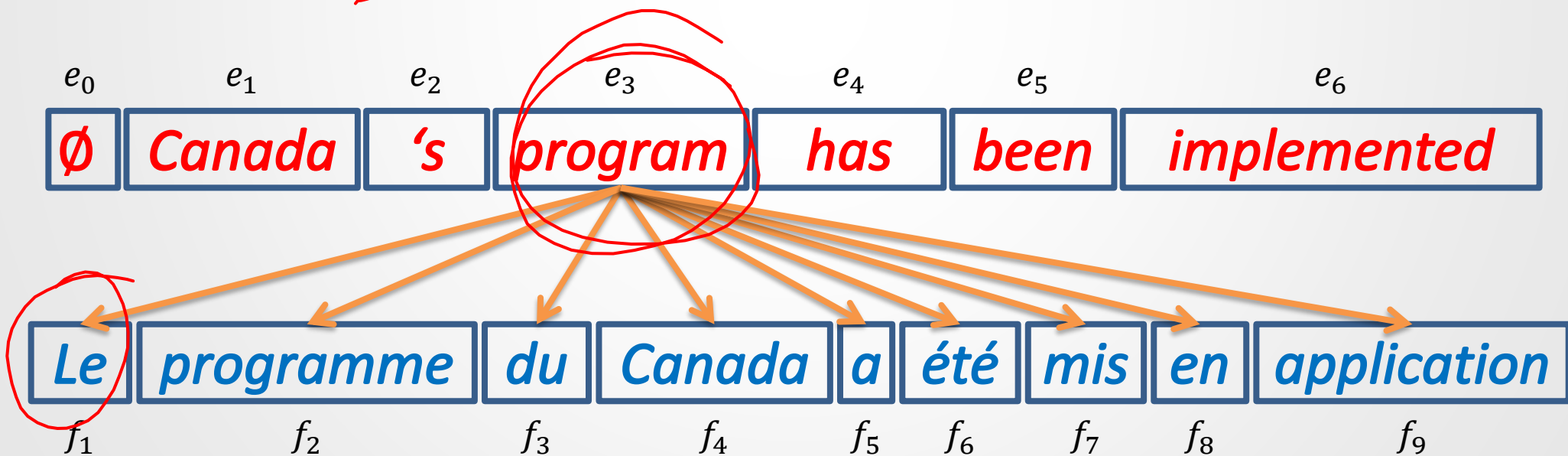
IBM Model 1: alignments



IBM Model 1: alignments

P(L|E) program

- There are $(L_E + 1) L_F$ possible alignments. (since $\|a\| = L_F$)
- IBM-1 doesn't know that some are *very bad* in reality.
 - E.g., $a = \{3, 3, 3, 3, 3, 3, 3, 3, 3\}$



IBM Model 1: alignments

- IBM Model 1 assumes that all alignments of E are **equally likely** given only the **length** (not the words) of F .

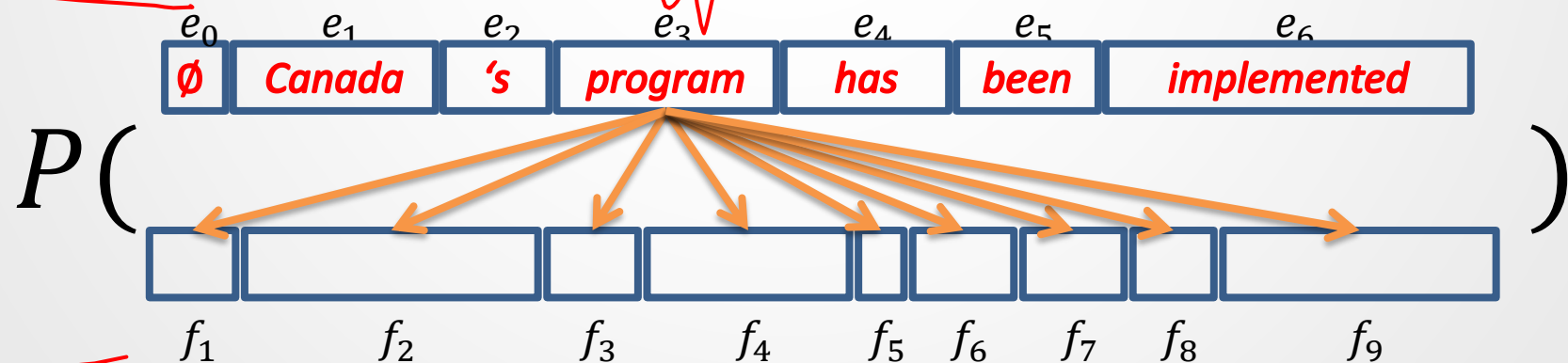
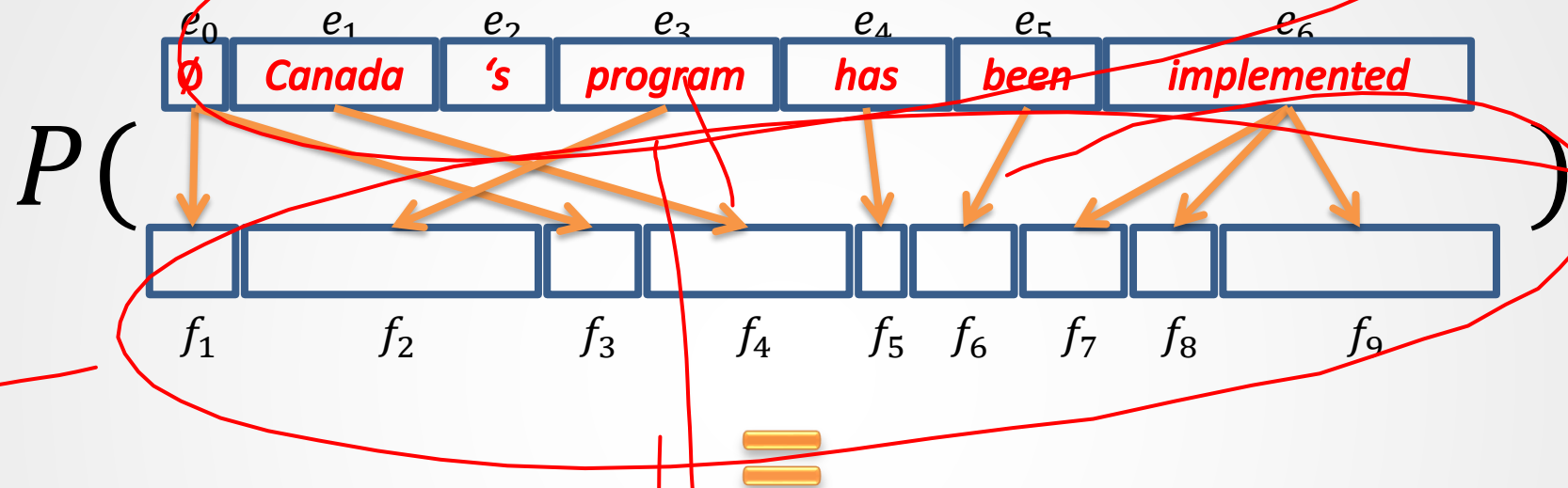
$$\forall a, P(a|E, L_F) = \frac{1}{(L_E + 1)^{L_F}}$$

Uniform over all possible alignments.

- This is a **major** simplifying assumption, but it gets the process started.

possible align.

Equally likely alignments *a priori*



IBM Model 1: translation probability

- Given an **alignment** a and an **English sentence** E , what is the probability of a **French sentence** F ?

$$P(F|a, E)$$

- In IBM-1,

$$P(F|a, E) = \prod_{j=1}^{L_F} P(f_j | e_{a_j})$$

(another simplifying assumption)

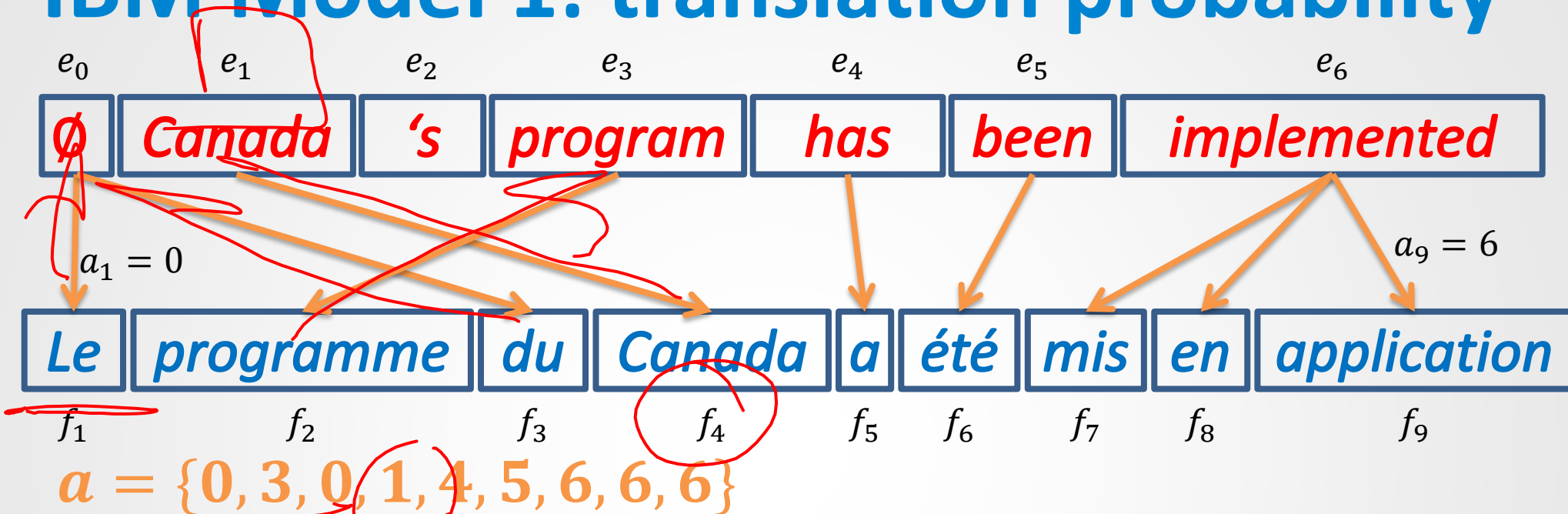
The probability of the j^{th} French word, given that it was generated from the a_j^{th} English word.

IBM Model 1: translation probability

- $E = \text{Canada's program has been implemented}$
- $a = \{0, 3, 0, 1, 4, 5, 6, 6, 6\}$
- $F = \text{Le programme du Canada à été mis en application}$

$$\begin{aligned} P(F|a, E) = & P(\text{Le}|\emptyset)P(\text{programme}|\text{program}) \times \\ & P(\text{du}|\emptyset)P(\text{Canada}|\text{Canada})P(\text{à}|\text{has}) \times \\ & P(\text{été}|\text{been})P(\text{mis}|\text{implemented}) \times \\ & P(\text{en}|\text{implemented}) \times \\ & P(\text{application}|\text{implemented}) \end{aligned}$$

IBM Model 1: translation probability



- $$P(F|a, E) = P(\text{Le}|\emptyset)P(\text{programme}|\text{program}) \times$$

$$P(\text{du}|\emptyset)P(\text{Canada}|\text{Canada})P(\text{à}|\text{has}) \times$$

$$P(\text{été}|\text{been})P(\text{mis}|\text{implemented}) \times$$

$$P(\text{en}|\text{implemented}) \times$$

$$P(\text{application}|\text{implemented})$$

IBM Model 1: generation

This is how we imagine English gets corrupted in the noisy channel.

- To generate a French sentence F from English E ,
 1. Pick a length of F (with probability $P(L_F)$).
 2. Pick an alignment (with uniform probability, $\frac{1}{(L_E + 1)^{L_F}}$).
 3. Sample French words with probability

Slide 17

$$P(F|a, E) = \prod_{j=1}^{L_F} P(f_j | e_{a_j})$$

Slide 19

So,

$$P(F, a | E) = P(a | E) P(F | a, E) = \frac{P(L_F)}{(L_E + 1)^{L_F}} \prod_{j=1}^{L_F} P(f_j | e_{a_j})$$

IBM-1: alignment as hidden variable

- If $P(F, a|E)$ describes the process of generating French words *and* alignments from English words...

- Then

Remember, the **noisy channel model** states that French words are really encoded English words!

$$P(F|E) = \sum_{a \in \mathcal{A}} P(F, a|E)$$

where \mathcal{A} is the **set of all possible** alignments

IBM-1: training

$$P(f|e)$$

- Our **training data** \mathcal{O} is a set of **pairs of corresponding** French and English sentences, $\mathcal{O} = \{(F_i, E_i)\}, i = 0..N$.
- If we knew the word alignments, a , learning $P(f|e)$ would be trivial with MLE: $P(f|e) = \frac{\text{Count}(f,e)}{\text{Count}(e)}$.
times aligned
- But the **alignments** are **hidden**. We need to use ...



IBM-1: expectation-maximization



- 1. Initialize** translation parameters $P(f|e)$ (e.g., randomly).
- 2. Expectation:** Given the current $\theta_k = P(f|e)$, compute the **expected value** of ~~$Count(f, e)$~~ for all words in training data \mathcal{O} .
- 3. Maximization:** Given the expected value of $Count(f, e)$, compute the **maximum likelihood** estimate of $\theta_k = P(f|e)$

$$P(f|e) = \frac{Count(f, e)}{Count(e)}$$

IBM-1 EM: Example

- Imagine our training data is
 $\mathcal{O} = \{(\textit{blue house}, \textit{maison bleue}), (\textit{the house}, \textit{la maison})\}$
- The vocabularies are
 $\mathcal{V}_E = \{\textit{blue}, \textit{house}, \textit{the}\}$ and
 $\mathcal{V}_F = \{\textit{maison}, \textit{bleue}, \textit{la}\}.$
- For **simplicity**, we consider only 1:1 alignments:
there is **no** NULL word, there are **no** zero-fertility words.

IBM-1 EM: Example

- First, we **initialize** our parameters, $\theta = P(f|e)$.
- In the **Expectation** step, we compute ~~expected~~ counts:
 - $TCount(f, e)$: the total number of times e and f are aligned.
 - $Total(e)$: the total number of e .

This has to be done in steps by first computing $P(F, a|E)$ then $P(a|F, E)$
- In the **Maximization** step, we perform MLE with the expected counts.

$$P(f|e) = \frac{\text{count}(f, e)}{\text{count}(e)}$$

IBM-1 EM: Example initialization

1. Make a table of $P(f|e)$ for all possible pairs f and e .

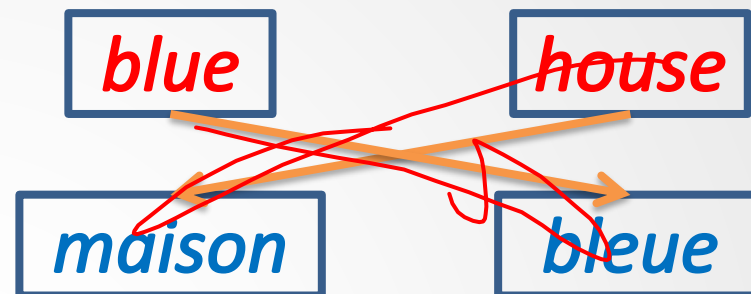
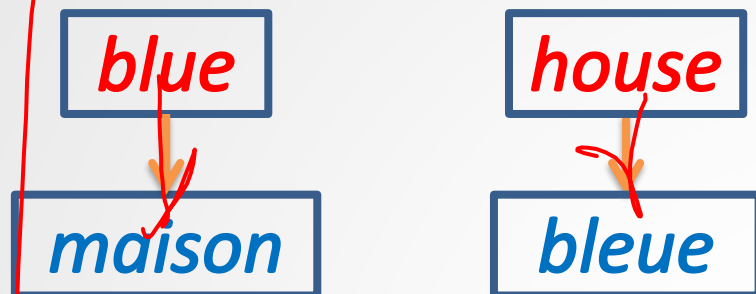
Initialize uniformly across rows.

θ_0 :

| | | |
|---|--|---|
| $P(\text{maison} \text{blue}) = \frac{1}{3}$ | $P(\text{bleue} \text{blue}) = \frac{1}{3}$ | $P(\text{la} \text{blue}) = \frac{1}{3}$ |
| $P(\text{maison} \text{house}) = \frac{1}{3}$ | $P(\text{bleue} \text{house}) = \frac{1}{3}$ | $P(\text{la} \text{house}) = \frac{1}{3}$ |
| $P(\text{maison} \text{the}) = \frac{1}{3}$ | $P(\text{bleue} \text{the}) = \frac{1}{3}$ | $P(\text{la} \text{the}) = \frac{1}{3}$ |

IBM-1 E: compute $P(F|a, E)$

'Sentence' 1



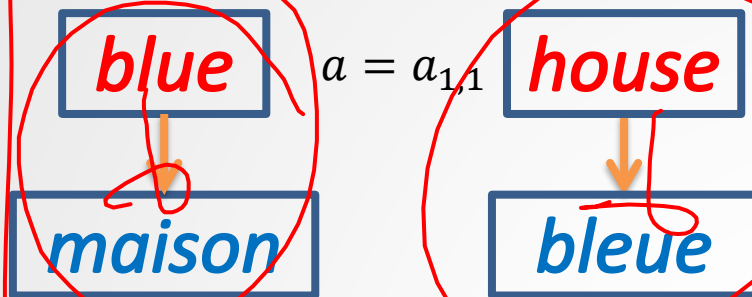
'Sentence' 2



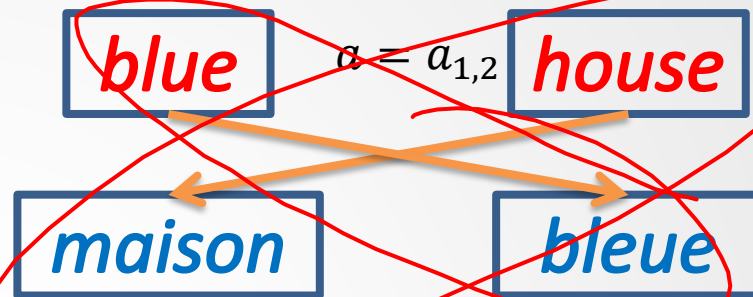
2. Make a grid where each sentence pair is a **row**, and each possible word-alignment is a **column**.

IBM-1 E: compute $P(F|a, E)$

'Sentence' 1

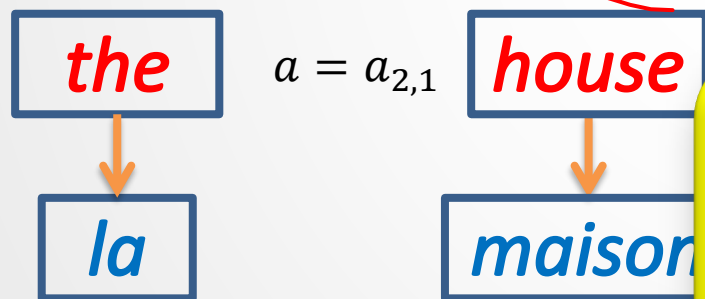


$$P(F|a, E) = P(\text{maison}|\text{blue}) \times P(\text{bleue}|\text{house}) = \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{9}$$



$$P(F|a, E) = P(\text{bleue}|\text{blue}) \times P(\text{maison}|\text{house}) = \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{9}$$

'Sentence' 2



$$P(F|a, E) = P(\text{la}|\text{the}) \times P(\text{maison}|\text{house}) = \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{9}$$

3. For each sentence pair and alignment, compute (slide 19)

$$P(F|a, E) = \prod_{f_j} P(f_j|e_{a_j})$$

IBM-1 E: compute $P(a|E, F)$

- We want the **probability of an alignment** a so that we can compute the **expected** $Count(f_j, e_i)$.

$$P(a|E, F) = \frac{P(F|a, E)}{\sum_{a_i \in \mathcal{A}} P(F|a_i, E)}$$

- This is **not** the same as the probability $P(a|E, L_F)$.
 - i.e., it won't *always* be uniform.

IBM-1 E: compute $P(a|E, F)$

$$P(a|E, F) = \frac{P(a, E, F)}{P(E, F)} = \frac{P(a, E, F)}{P(E)P(F|E)} = \frac{P(F, a|E)P(E)}{P(E)P(F|E)}$$

$$\stackrel{(*)}{=} \frac{P(F, a|E)}{\sum_{a_i \in \mathcal{A}} P(F, a_i|E)} \stackrel{(**)}{=} \frac{P(F|a, E)}{\sum_{a_i \in \mathcal{A}} P(F|a_i, E)}$$

(*) Because $P(F|E) = \sum_{a_i \in \mathcal{A}} P(F, a_i|E)$ (slide 23)

(**) Rewrite $P(F, a|E)$ as on slide 22,
and $\frac{P(L_F)}{(L_E+1)^{L_F}}$ cancels out

IBM-1 E: compute $P(a|E, F)$

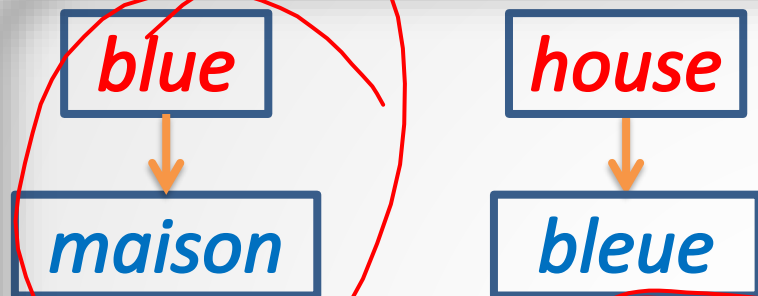
- We want the **probability of an alignment** a so that we can compute the **expected** $Count(f_j, e_i)$.

$$P(a|E, F) = \frac{P(F|a, E)}{\sum_{a_i \in \mathcal{A}} P(F|a_i, E)}$$

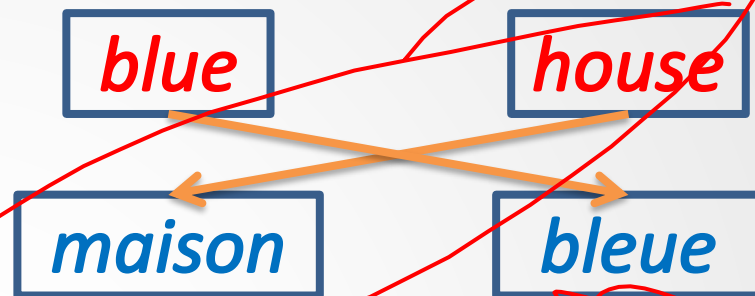
- This is **not** the same as the probability $P(a|E, L_F)$.
 - i.e., it won't *always* be uniform.

IBM-1 E: compute $P(a|E, F)$

'Sentence' 1

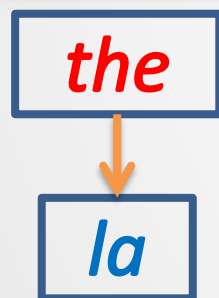


$$P(a|E, F) = \frac{1/9}{1/9 + 1/9} = \frac{1}{2}$$

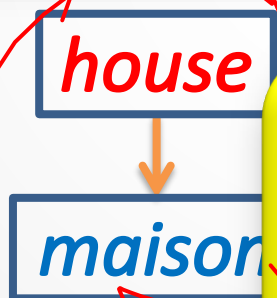


$$P(a|E, F) = \frac{1/9}{1/9 + 1/9} = \frac{1}{2}$$

'Sentence' 2



$$P(a|E, F) = \frac{1/9}{1/9 + 1/9} = \frac{1}{2}$$



4. For each element in your grid, divide $P(F|a, E)$ by the sum of the row (slide 33).

IBM-1 E: compute *TCount*

maison and *blue* are aligned only in alignment 1, sentence 1.

$$P(a = 1|F_1, E_1) = \frac{1}{2}$$

maison and *house* are aligned in alignment 2, sentence 1.
and
alignment 1, sentence 2

$$P(a = 2|F_1, E_1) = \frac{1}{2}$$

$$P(a = 1|F_2, E_2) = \frac{1}{2}$$

| | |
|---|---|
| <i>TCount(maison, blue)</i> $= \frac{1}{2}$ | <i>TCount(bleue, blue)</i> $= \frac{1}{2}$ |
| <i>TCount(maison, house)</i> $= \frac{1}{2} + \frac{1}{2} = 1$ | <i>TCount(bleue, house)</i> $= ?$ |
| <i>TCount(maison, the)</i> $= ?$ | <i>TCount(bleue, the)</i> $= ?$ |

- For each possible word pair e and f , sum $P(a|E, F)$ from step 4 across all alignments and sentence pairs for each instance that e is aligned with f

This is a **new** table,
not the $\theta = P(f|e)$ table
from before!

IBM-1 E: compute *TCount*

→ # of times
e aligned f

| | | |
|--|---|--|
| $TCount(maison, blue)$ $= \frac{1}{2}$ | $TCount(bleue, blue)$ $= \frac{1}{2}$ | $TCount(la, blue)$ $= 0$ |
| $TCount(maison, house)$ $= \frac{1}{2} + \frac{1}{2} = 1$ | $TCount(bleue, house)$ $= \frac{1}{2}$ | $TCount(la, house)$ $= \frac{1}{2}$ |
| $TCount(maison, the)$ $= \frac{1}{2}$ | $TCount(bleue, the)$ $= 0$ | $TCount(la, the) = \frac{1}{2}$ |

IBM-1 E: compute *Total*

E.g.,

$$\text{Total}(\text{blue}) = \frac{1}{2} + \frac{1}{2} = 1,$$

$$\text{Total}(\text{house}) = 1 + \frac{1}{2} + \frac{1}{2} = 2, \quad \dots$$

$$\begin{array}{l} T\text{Count}(\text{maison}, \text{blue}) \\ = \frac{1}{2} \end{array}$$

$$\begin{array}{l} T\text{Count}(\text{bleue}, \text{blue}) \\ = \frac{1}{2} \end{array}$$

$$\begin{array}{l} T\text{Count}(\text{la}, \text{blue}) \\ = 0 \end{array}$$

$$\begin{array}{l} T\text{Count}(\text{maison}, \text{house}) \\ = \frac{1}{2} + \frac{1}{2} = 1 \end{array}$$

$$\begin{array}{l} T\text{Count}(\text{bleue}, \text{house}) \\ = \frac{1}{2} \end{array}$$

$$\begin{array}{l} T\text{Count}(\text{la}, \text{house}) \\ = \frac{1}{2} \end{array}$$

$$\begin{array}{l} T\text{Count}(\text{maison}, \text{the}) \\ = \frac{1}{2} \end{array}$$

6. Sum over the rows of this table to get the **total** estimates for **each** English word, *e*.

IBM-1 M: Recompute $P(f|e)$

7. Compute $P(f|e) = \frac{TCount(f,e)}{Total(e)}$

This is your model after iteration 1.

θ_1 :

| | | |
|---------------------------------------|---|--|
| $P(maison blue)$ $= \frac{1/2}{1}$ | $P(bleue blue)$ $= \frac{1/2}{1}$ | $P(la blue)$ $= \frac{0}{1}$ |
| $P(maison house)$ $= \frac{1}{2}$ | $P(bleue house)$ $= \frac{1/2}{2} = \frac{1}{4}$ | $P(la house)$ $= \frac{1/2}{2} = \frac{1}{4}$ |
| $P(maison the)$ $= \frac{1/2}{1}$ | $P(bleue the)$ $= \frac{0}{1}$ | $P(la the)$ $= \frac{1/2}{1}$ |

IBM-1 EM: Repeat

- You have finished **1 iteration** of EM when you have completed Step 7,
- Go back to Step 2 and repeat.

IBM-1 EM: Repeat

1. Initialize $P(f|e)$

2. Make grid of all possible alignments

3. Compute $P(F|a, E) \rightarrow$ Products of $P(f|e)$

4. Compute $P(a|E, F) \rightarrow$ Divide by sum of rows from step 3

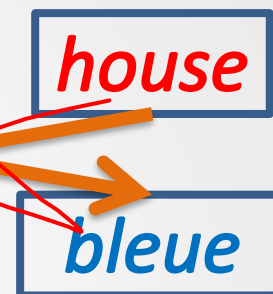
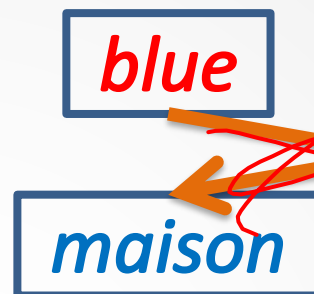
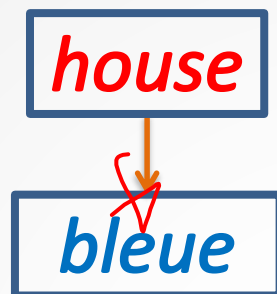
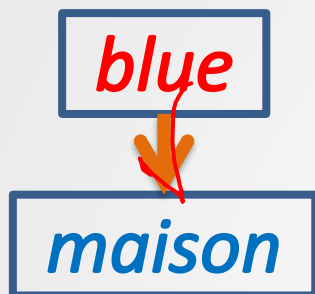
5. Compute $TCount \rightarrow$ Sum relevant probabilities from step 4

6. Compute $Total \rightarrow$ Sum over rows from step 5

7. Compute $P(f|e) = \frac{TCount(f,e)}{Total(e)}$

IBM-1 E: compute $P(F|a, E)$

'Sentence' 1



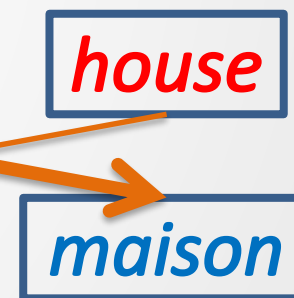
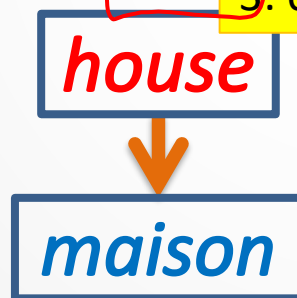
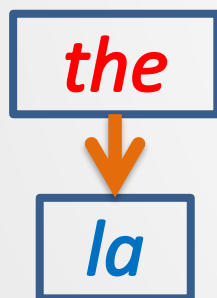
$$P(F|a, E) = P(\text{maison}|\text{blue}) \times P(\text{bleue}|\text{house}) = \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$$

$$P(F|a, E) = P(\text{bleue}|\text{blue}) \times P(\text{maison}|\text{house}) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

2: make grid

3: compute products of $P(f|e)$

'Sentence' 2

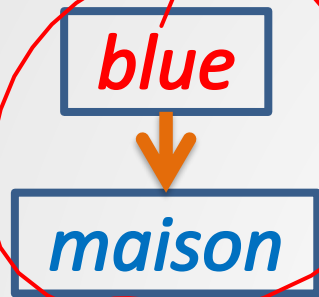


$$P(F|a, E) = P(\text{la}|\text{the}) \times P(\text{maison}|\text{house}) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

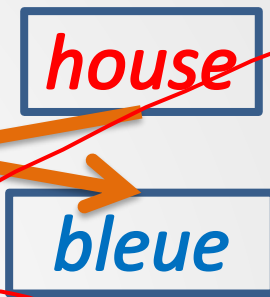
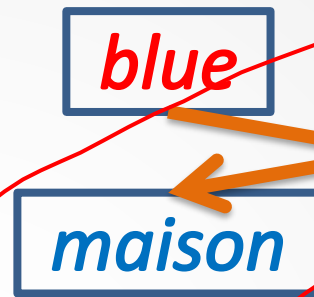
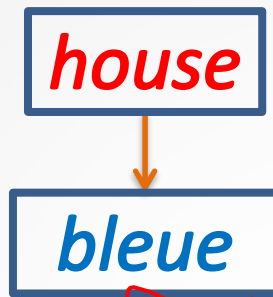
$$P(F|a, E) = P(\text{maison}|\text{the}) \times P(\text{la}|\text{house}) = \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$$

IBM-1 E: compute $P(a|E, F)$

'Sentence' 1



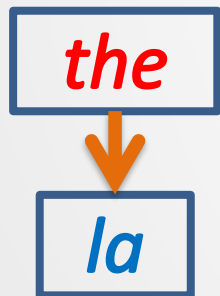
$$P(a|E, F) = \frac{1/8}{1/8 + 1/4} = \frac{1}{3}$$



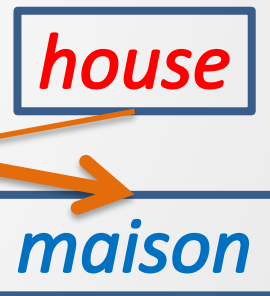
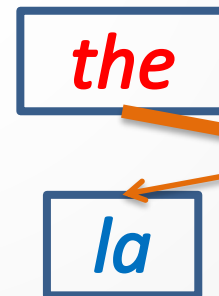
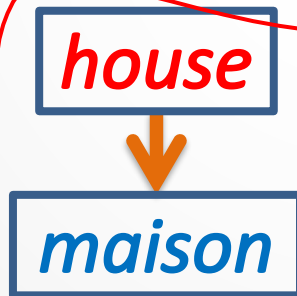
$$P(a|E, F) = \frac{1/4}{1/8 + 1/4} = \frac{2}{3}$$

4: divide by sum of rows in step 3

'Sentence' 2



$$P(a|E, F) = \frac{1/4}{1/4 + 1/8} = \frac{2}{3}$$



$$P(a|E, F) = \frac{1/8}{1/4 + 1/8} = \frac{1}{3}$$

IBM-1 E: compute *TCount* & *Total*

$$\text{Total}(\text{blue}) = \frac{1}{3} + \frac{2}{3} = 1, \quad \text{Total}(\text{house}) = \frac{4}{3} + \frac{1}{3} + \frac{1}{3} = 2,$$

$$\text{Total}(\text{the}) = \frac{1}{3} + \frac{2}{3} = 1$$

5. Compute *TCount* by summing relevant probabilities from step 4
6. Compute *Total* by summing rows

| | | |
|--|---|--|
| $TCount(\text{maison}, \text{blue})$ $= \frac{1}{3}$ | $TCount(\text{bleue}, \text{blue})$ $= \frac{2}{3}$ | $TCount(\text{la}, \text{blue})$ $= 0$ |
| $TCount(\text{maison}, \text{house})$ $= \frac{2}{3} + \frac{2}{3} = \frac{4}{3}$ | $TCount(\text{bleue}, \text{house})$ $= \frac{1}{3}$ | $TCount(\text{la}, \text{house})$ $= \frac{1}{3}$ |
| $TCount(\text{maison}, \text{the})$ $= \frac{1}{3}$ | $TCount(\text{bleue}, \text{the})$ $= 0$ | $TCount(\text{la}, \text{the})$ $= \frac{2}{3}$ |

maison

bleue = blue

IBM-1 M: Recompute $P(f|e)$

↪ house

- Compute $P(f|e) = \frac{TCount(f,e)}{Total(e)}$
- θ_2 :


Ties have been broken

e.g.,

$$P(\text{maison}|\text{blue}) \neq P(\text{bleue}|\text{blue})$$

| | | |
|--|---|--|
| $P(\text{maison} \text{blue})$ $= \frac{1/3}{1}$ | $P(\text{bleue} \text{blue})$ $= \frac{2/3}{1}$ | $P(\text{la} \text{blue})$ $= \frac{0}{1}$ |
| $P(\text{maison} \text{house})$ $= \frac{4/3}{2} = \frac{2}{3}$ | $P(\text{bleue} \text{house})$ $= \frac{1/3}{2} = \frac{1}{6}$ | $P(\text{la} \text{house})$ $= \frac{1/3}{2} = \frac{1}{6}$ |
| $P(\text{maison} \text{the})$ $= \frac{1/3}{1}$ | $P(\text{bleue} \text{the})$ $= \frac{0}{1}$ | $P(\text{la} \text{the})$ $= \frac{2/3}{1}$ |

Practical note on programming IBM-1

- If you were to code the EM algorithm for IBM-1, you would **not** initialize $\theta = P(f|e)$ uniformly over the **entire** vocabulary.
 - Don't make a $V_F \times V_E$ table with $P(f|e) = 1/||V_E||$ 
- This structure would be too **large**.
 - Probabilities would be too **small**.
 - It would take **too much work** to update.
- Rather, initialize a hash table over possible alignments, \mathcal{M} . For every English word e , only consider French words f in sentences **aligned** with English sentences containing e .
 - e.g., structure $P.e.f := P(f|e) = 1/||\mathcal{M}||$

Higher IBM models

| | |
|-------------|--|
| IBM Model 1 | lexical translation |
| IBM Model 2 | adds absolute re-ordering model |
| IBM Model 3 | adds fertility model |
| ... | ... |

- Only IBM Model 1 training reaches a *global maximum*
 - Training of each IBM model **extends** the **next lowest** model.
- **Higher models** become computationally **expensive**.

IBM-2

- Unlike IBM Model-1, the placement of a word in, say, **Spanish** in **IBM Model-2** depends on where its **equivalent** word was in **English**.
 - IBM-2 captures the intuition that translations should lie roughly “along the diagonal”.



| | Buenos | dias | , | me | gusta | papas | frías |
|----------|--------|------|---|----|-------|-------|-------|
| Good | X | | | | | | |
| day | | X | | | | | |
| , | | | X | | | | |
| I | | | | X | | | |
| like | | | | | X | | |
| cold | | | | | | | X |
| potatoes | | | | | | X | |

IBM-2

- IBM Model 2 **builds** on Model 1 by adding a **re-ordering model** defined by **distortion** parameters regardless of actual words.

$D(i|j, \mathcal{L}_E, \mathcal{L}_F)$ = the probability that the i^{th} **English slot** is aligned to the j^{th} **French slot**, given sentence lengths \mathcal{L}_E and \mathcal{L}_F .

- In IBM Model 2:

$$P(a|E, \mathcal{L}_E, \mathcal{L}_F) = \prod_{j=1}^{\mathcal{L}_F} D(a_j|j, \mathcal{L}_E, \mathcal{L}_F)$$

- Recall that in IBM Model 1,

$$P(a|E, \mathcal{L}_E, \mathcal{L}_F) = \frac{P(\mathcal{L}_F)}{(\mathcal{L}_E + 1)^{\mathcal{L}_F}}$$

IBM-2 – Probability of alignment

- $E = \text{And the program has been implemented}$

- $F = \text{Le programme a été mis en application}$

- $\mathcal{L}_E = 6$

- $\mathcal{L}_F = 7$

- $a = \{2, 3, 4, 5, 6, 6, 6\}$ (i.e., $f_1 \leftarrow e_2, f_2 \leftarrow e_3, \dots$)

$D(\text{2nd English word} | \text{1st French word}, \dots)$

$$P(a|E, \mathcal{L}_E, \mathcal{L}_F) = D(2|1, 6, 7) \times D(3|2, 6, 7) \times D(4|3, 6, 7) \times D(5|4, 6, 7) \times D(6|5, 6, 7) \times D(6|6, 6, 7) \times D(6|7, 6, 7)$$

This is independent of the actual words.

This cares only about position.

IBM-2: generation

- To **generate** a French sentence F from English E ,

1. **Pick an alignment** with probability

$$\prod_{j=1}^{\mathcal{L}_F} D(a_j | j, \mathcal{L}_E, \mathcal{L}_F)$$

3. **Sample** French words with probability

$$P(F | a, E) = \prod_{j=1}^{\mathcal{L}_F} P(f_j | e_{a_j})$$

This is the same $P(f|e)$ as in IBM-1.

So,

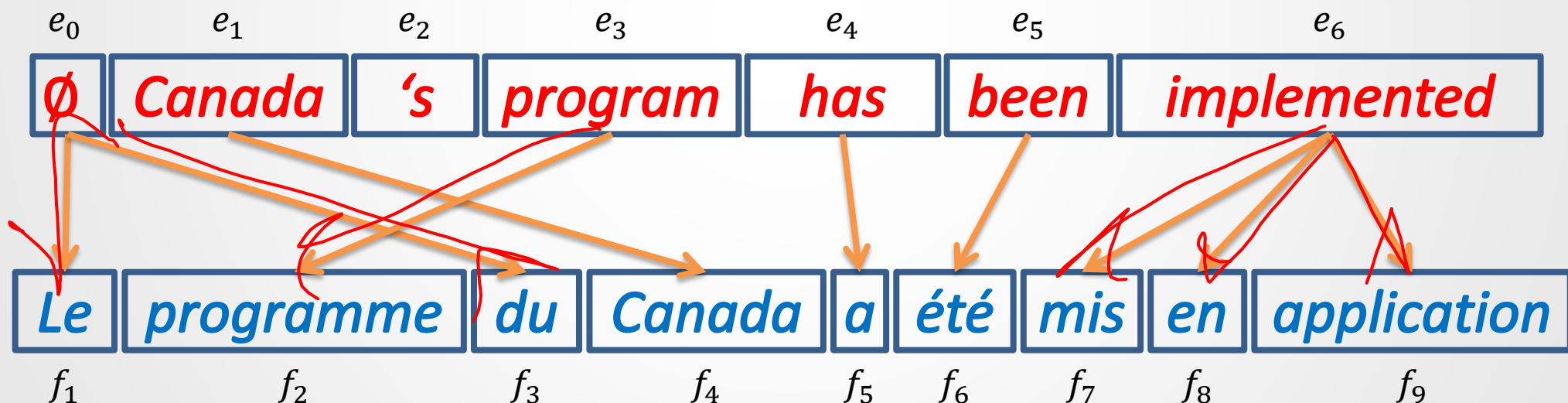
$$P(F, a | E) = P(a | E) P(F | a, E) = \prod_{j=1}^{\mathcal{L}_F} D(a_j | j, \mathcal{L}_E, \mathcal{L}_F) P(f_j | e_{a_j})$$

IBM-2: training

- We use EM, as before with IBM-1 **except** that we need to take the **distortion** into account when computing the probability of an alignment.
- We also need to **learn** the distortion function.
- *Aren't you glad that you don't need to know how to compute EM for IBM-2?*

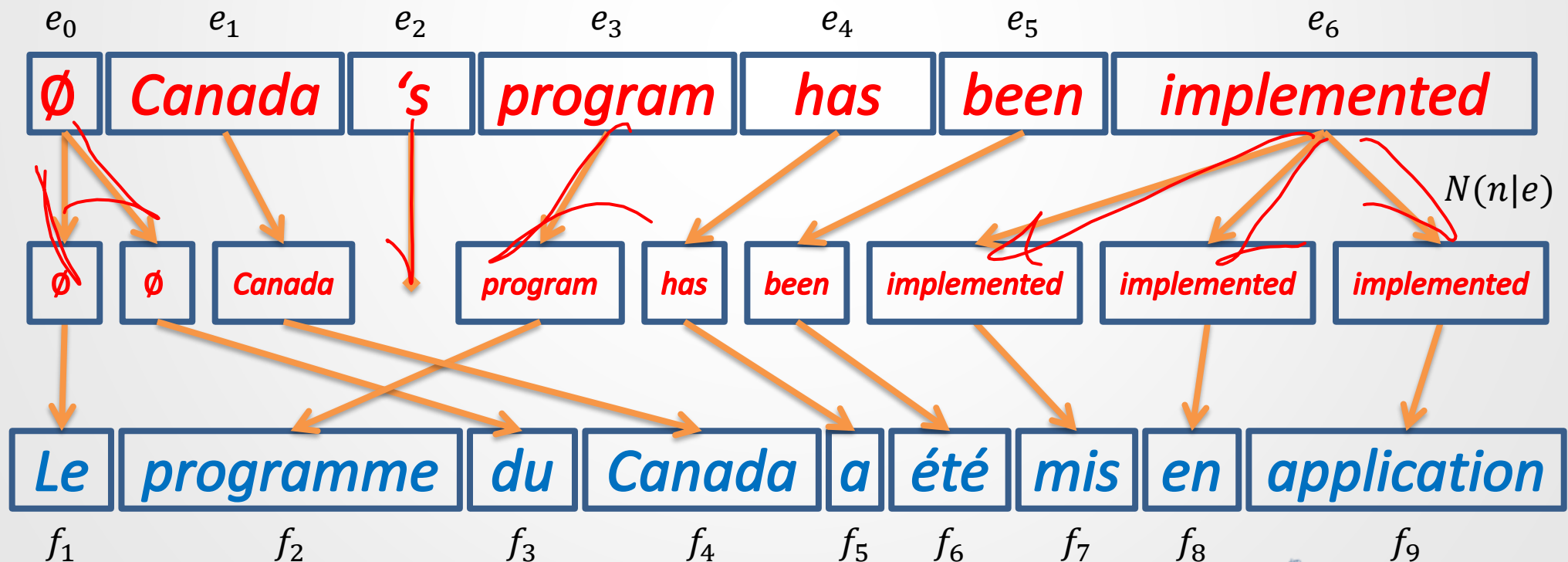
IBM-3

- **IBM Model 3** extends Model 2 by adding a **fertility model** that describes how many **French** words each **English** word can produce.
 - In the example below, implemented appears to be **more fertile** than *program*.



IBM-3: The generation model

- First, we **replicate** each word according to a new hidden parameter, $N(n|e)$, which is the **probability that word e produces n words**.
- We then **re-align** (*with distortion*) and **translate** as we did in IBM-2.

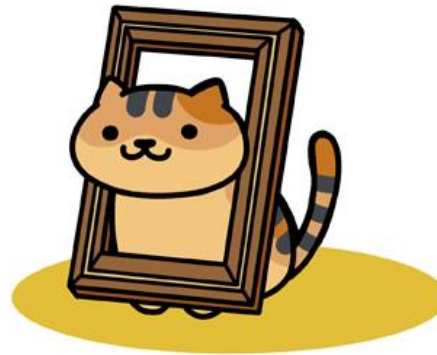


IBM models

| | |
|-------------|---------------------------------|
| IBM Model 1 | lexical translation |
| IBM Model 2 | adds absolute re-ordering model |
| IBM Model 3 | adds fertility model |

Reading

- **Entirely optional:** Vogel, S., Ney, H., and Tillman, C. (1996). *HMM-based Word Alignment in Statistical Translation*. In: Proceedings of the 16th International Conference on Computational Linguistics, pp. 836-841, Copenhagen.
- (optional) Gale & Church “*A Program for Aligning Sentences in Bilingual Corpora*” (on course website)
- **Useful reading on IBM Model-1:** Section 25.5 of the 2nd edition of the Jurafsky & Martin text.
 - 1st edition available at Robarts library.
- **Other:** Manning & Schütze Sections 13.0, 13.1.2 (Gale&Church), 13.1.3 (Church), 13.2, 13.3, 14.2.2



Quote by Vincent van Gogh
I am always doing what I
cannot do yet, in order to
learn how to do it

