CS60075
Natural Language Processing
Autumn 2020

Module 7:

Machine Translation 2

16 October 2020

Lexical Translation

- Goal: A model $p(\boldsymbol{e}|\boldsymbol{f}, m)$

$$\boldsymbol{e} = \langle e_1, e_2, \dots, e_m \rangle$$
$$\boldsymbol{f} = \langle f_1, f_2, \dots, f_n \rangle$$

- Assumptions
  - Each word e_i in e is generated from exactly one word in f
  - Thus, we have an alignment a_i that indicates which word e_i came from. f_{a_i}
  - Given the alignments a, translation decisions are conditionally independent of each other and depend only on the aligned source word f_{a_i}

Lexical Translation

$$p(\boldsymbol{e}|\boldsymbol{f}, m) = \sum_{a \in [0,n]^m} p(\boldsymbol{a}|\boldsymbol{f}, m) \times \prod_{i=1}^{m} p(e_i|f_{a_i})$$

Alignment × Translation | Alignment

IBM Model 1: P(E|F)

- Translation probability
  - For a foreign sentence $f = \left(f_1, \ldots, f_{l_f}\right)$ of length l_f
  - To an English sentence e $= \left(e_1, \ldots, e_{l_e}\right)$ of length l_e
  - With an alignment of each English word e_j to a foreign word f_i according to the alignment function $a: j \to i$

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

Computing P(E|F) in IBM Model 1

$p(a|f)$

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

- A normalization factor, since there are $(l_f + 1)^{l_e}$ possible alignments
- Parameter ϵ is a normalization constant
- The probability of an alignment given the foreign sentence

Computing P(E|F) in IBM Model 1

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

$$p(\boldsymbol{e}|\boldsymbol{f}) = \sum_{a} p(\boldsymbol{e},\boldsymbol{a}|\boldsymbol{f}) = \sum_{a} p(\boldsymbol{a}|\boldsymbol{f}) \times \prod_{j=1}^{l_e} p\left(e_j|f_{a_j}\right)$$

Example

das

e	$t(e\|f)$
the	0.7
that	0.15
which	0.075
who	0.05
this	0.025

Haus

e	$t(e\|f)$
house	0.8
building	0.16
home	0.02
household	0.015
shell	0.005

ist

e	$t(e\|f)$
is	0.8
's	0.16
exists	0.02
has	0.015
are	0.005

klein

e	$t(e\|f)$
small	0.4
little	0.4
short	0.1
minor	0.06
petty	0.04

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

Estimate Translation Probabilities

$$\hat{p}_{\text{MLE}}(e \mid \texttt{Haus}) = \begin{cases} 0.8 & \text{if } e = \text{house}, \\ 0.16 & \text{if } e = \text{building}, \\ 0.02 & \text{if } e = \text{home}, \\ 0.015 & \text{if } e = \text{household}, \\ 0.005 & \text{if } e = \text{shell}. \end{cases}$$

Estimate Alignments Given t-table

If we have translation probabilities

das

e	$t(e\|f)$
the	0.7
that	0.15
which	0.075
who	0.05
this	0.025

Haus

e	$t(e\|f)$
house	0.8
building	0.16
home	0.02
household	0.015
shell	0.005

ist

e	$t(e\|f)$
is	0.8
's	0.16
exists	0.02
has	0.015
are	0.005

klein

e	$t(e\|f)$
small	0.4
little	0.4
short	0.1
minor	0.06
petty	0.04

The goal is to find the most probable alignment given a parameterized model

Estimating the Alignment

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

$$\boldsymbol{a}^* = argmax_{\boldsymbol{a}}\, p(\boldsymbol{e}, \boldsymbol{a} | \boldsymbol{f})$$

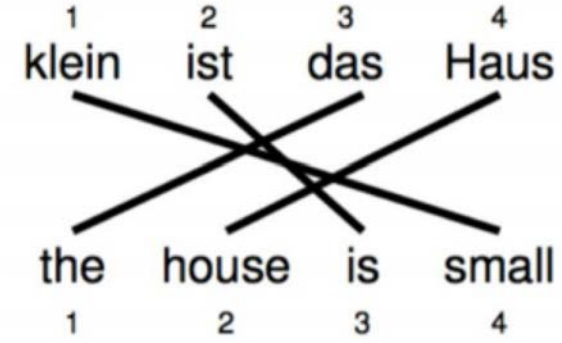
$$= argmax_{\boldsymbol{a}} \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j | f_{a(j)})$$

$$= argmax_a \prod_{j=1}^{l_e} t(e_j | f_{a(j)})$$

Since translation choice for each position is independent, the product is maximized by maximizing each term:

$$a_i{}^* = argmax_{a_i = 0}^{n}\; t(e_i | f_{a_i})$$

Learning Lexical Translation Models



- We'd like to estimate the lexical translation probabilities $t(e|f)$ from a parallel corpus but we do not have the alignments
- **Chicken and egg problem**
  - If we had the alignments, we could estimate the parameters of our generative model (MLE)
  - If we had the parameters, we could estimate the alignments

klein

e	$t(e\|f)$

klein

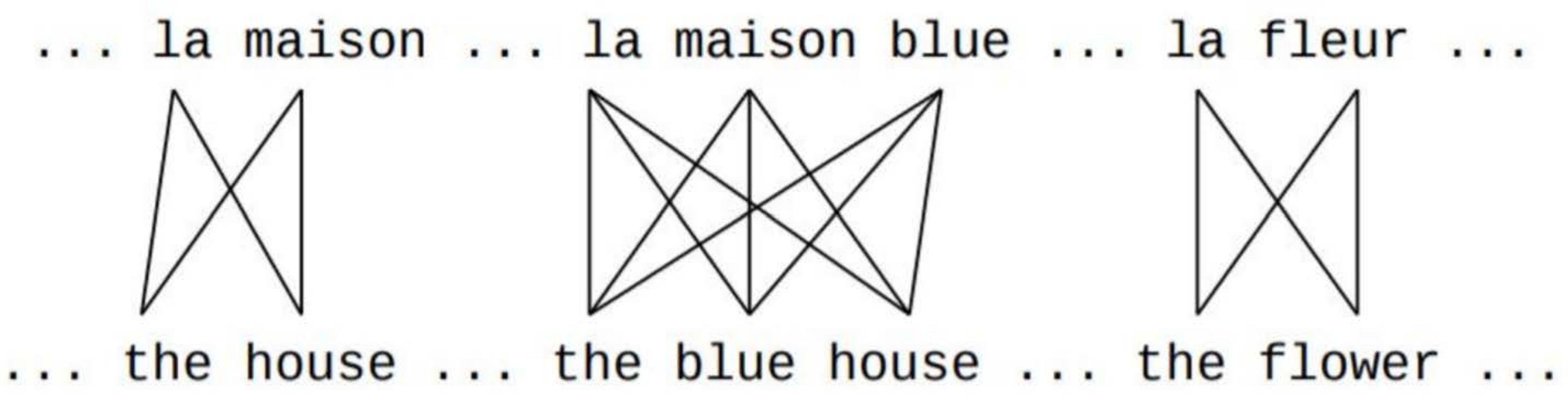
e	$t(e\|f)$
small	0.4
little	0.4
short	0.1
minor	0.06
petty	0.04

EM Algorithm

- Incomplete data
  - If we had complete data, we could estimate the model
  - If we had the model, we could fill in the gaps in the data

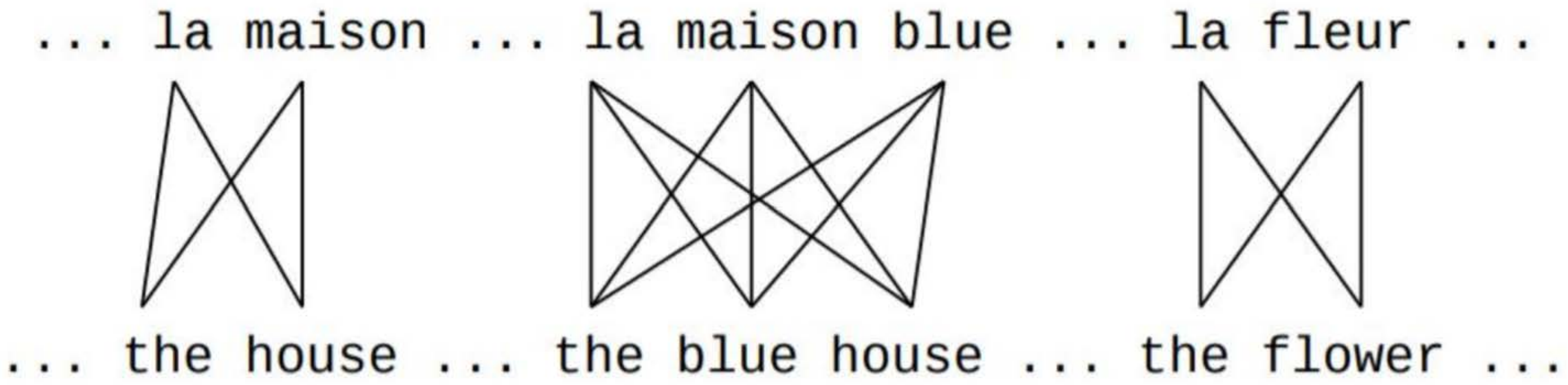
- **Expectation Maximization (EM)** in a nutshell
  1. Initialize model parameters (e.g., uniform, random)
  2. Assign probabilities to the missing data
  3. Estimate model parameters from completed data
  4. Iterate steps 2-3 until convergence

EM Algorithm



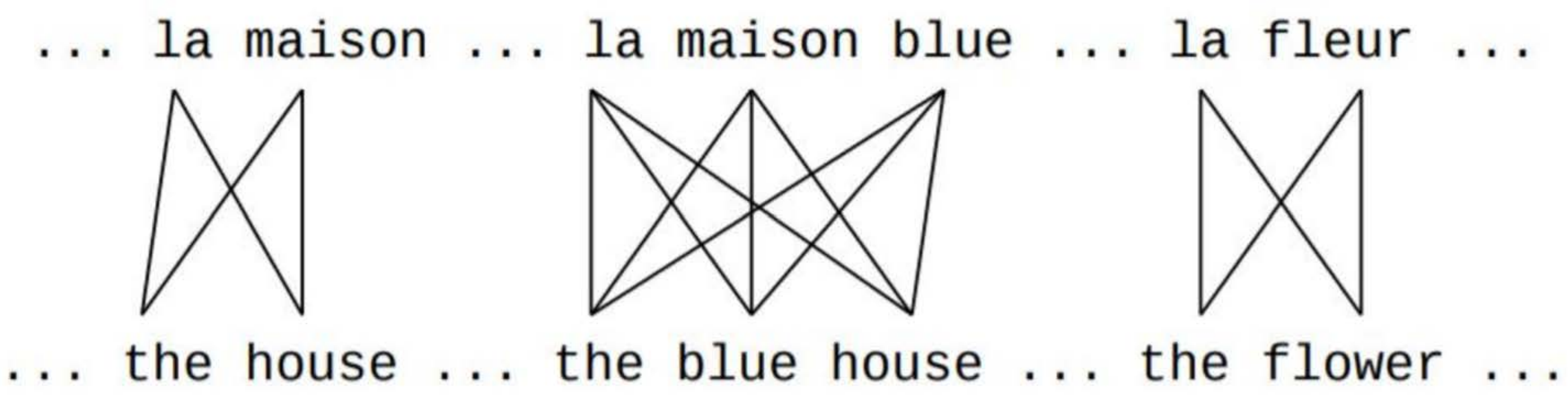
- Initial step: all word alignments equally likely
- Model learns that: e.g., *la* is often aligned with *the*

EM Algorithm



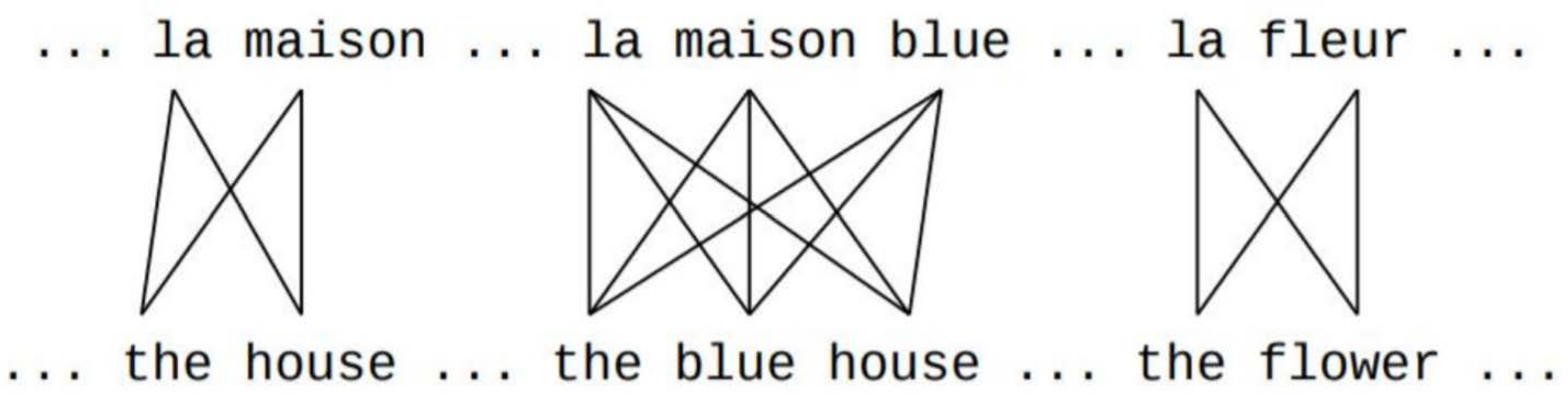
- After one iteration
- Alignments, e.g., between la and the are more likely

EM Algorithm



- After another iteration
- Alignments, e.g., between *fleur* and *flower* are more likely

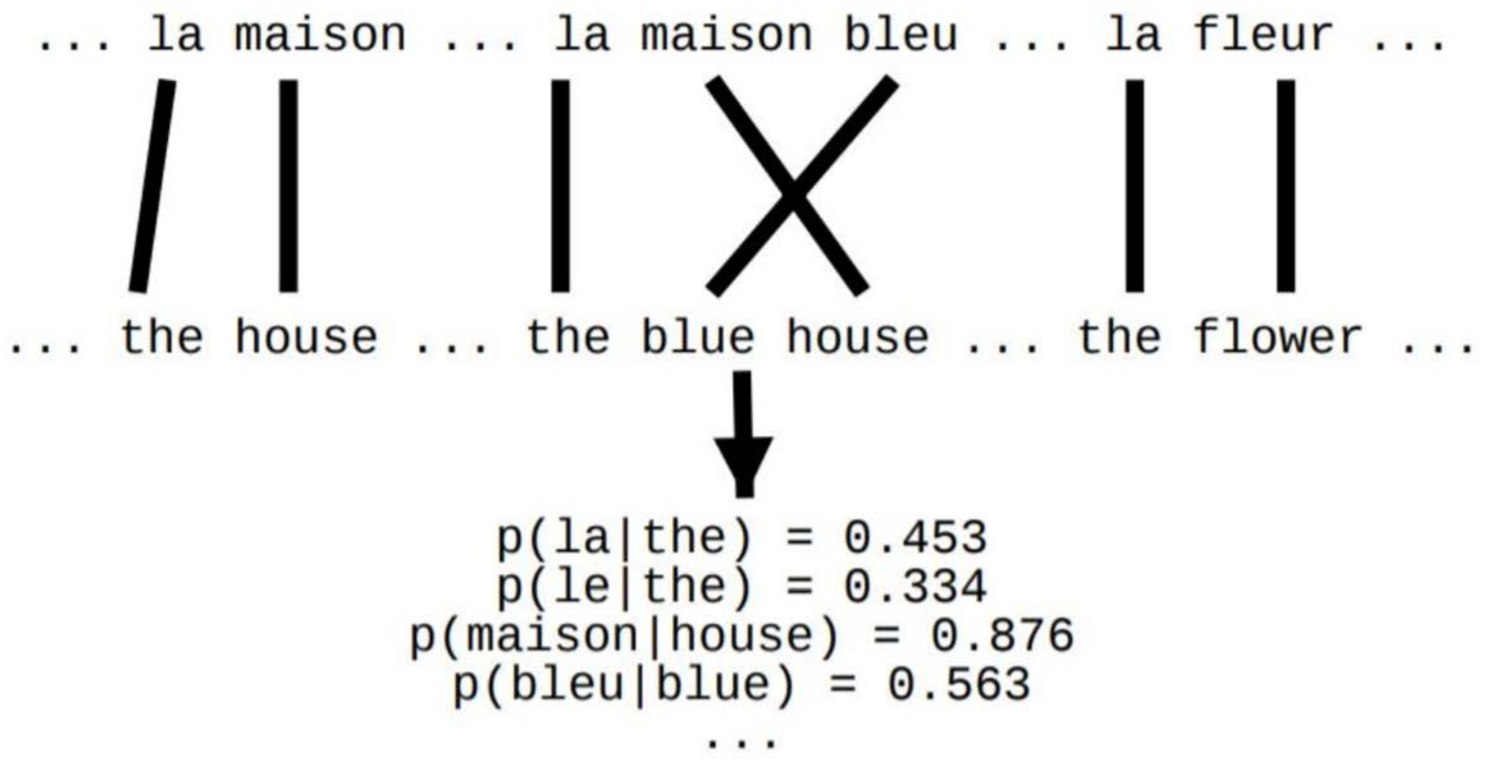
EM Algorithm



- Convergence
- Inherent hidden structure revealed by EM

EM Algorithm:

Parameter estimation from the aligned corpus



The EM Algorithm for Word Alignment

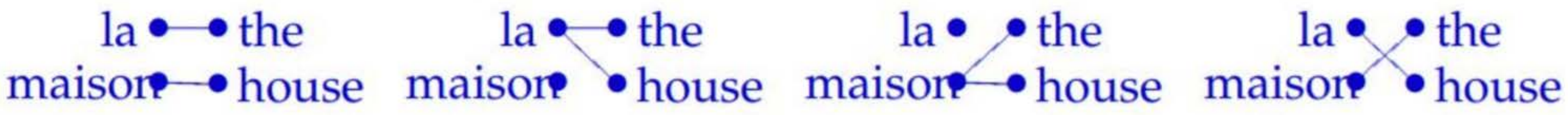
- Initialize the mode, typically with uniform distributions
- Repeat
  - **E Step**: use the current model to compute the probability of all possible alignments of the training data
  - **M Step**: use these alignment probability estimates to re-estimate values for all of the parameters
- Until convergence (i.e., parameters no longer change)

IBM Model 1 and EM

t-table **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



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

IBM Model 1 and EM: Expectation Step

$$p(a\,|\boldsymbol{e},\boldsymbol{f}) = \frac{p(\boldsymbol{e},a|\,\boldsymbol{f})}{p(\boldsymbol{e}|\boldsymbol{f})}$$

$$= \frac{\frac{\epsilon}{\left(l_f+1\right)^{l_e}}\prod_{j=1}^{l_e} t(e_j|f_{a_j})}{\frac{\epsilon}{\left(l_f+1\right)^{l_e}}\prod_{j=1}^{l_e}\sum_{i=0}^{l_f} t(e_j|f_{a_j})}$$

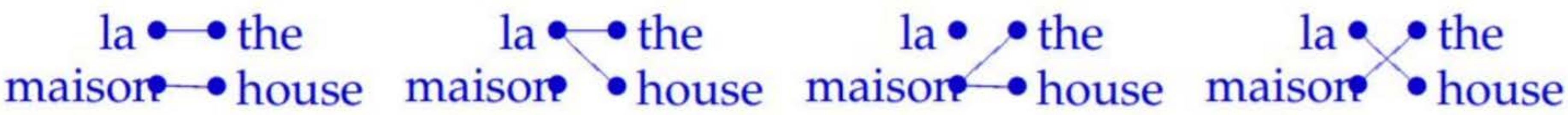
$$= \prod_{j=1}^{l_e}\frac{t(e_j|f_{a_j})}{\sum_{i=0}^{l_f} t(e_j|f_{a_j})}$$

IBM Model 1 and EM

t-table **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



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

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

IBM Model 1 and EM: Maximization Step

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

$$c(e|f) = \sum_{a} p(a|\boldsymbol{e}, \boldsymbol{f}) \sum_{j=1}^{l_e} \delta(e, e_j)\delta(f, f_{a_j})$$

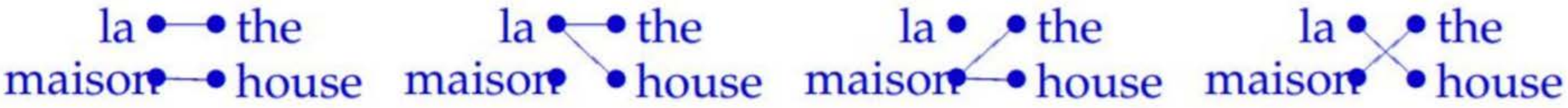
IBM Model 1 and EM: Maximization Step

t-table **Probabilities**

$p(\text{the}|\text{la}) = 0.7$ $\quad p(\text{house}|\text{la}) = 0.05$

$p(\text{the}|\text{maison}) = 0.1$ $\quad p(\text{house}|\text{maison}) = 0.8$

Alignments



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

E-step $p(a|\mathbf{e}, \mathbf{f}) = 0.824$ $\quad p(a|\mathbf{e}, \mathbf{f}) = 0.052$ $\quad p(a|\mathbf{e}, \mathbf{f}) = 0.118$ $\quad p(a|\mathbf{e}, \mathbf{f}) = 0.007$

M-step **Counts**

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

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

IBM Model 1 and EM: Maximization Step

t-table **Probabilities**

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

E-step **Alignments**

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

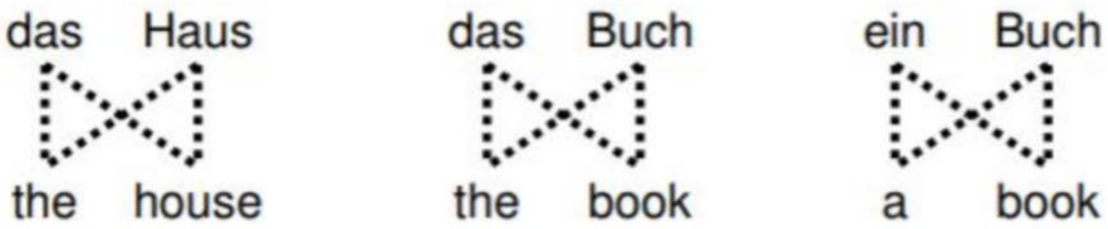
M-step **Counts**

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

Update t-table:

$$p(\text{the}|\text{la}) = c(\text{the}|\text{la})/c(\text{la})$$

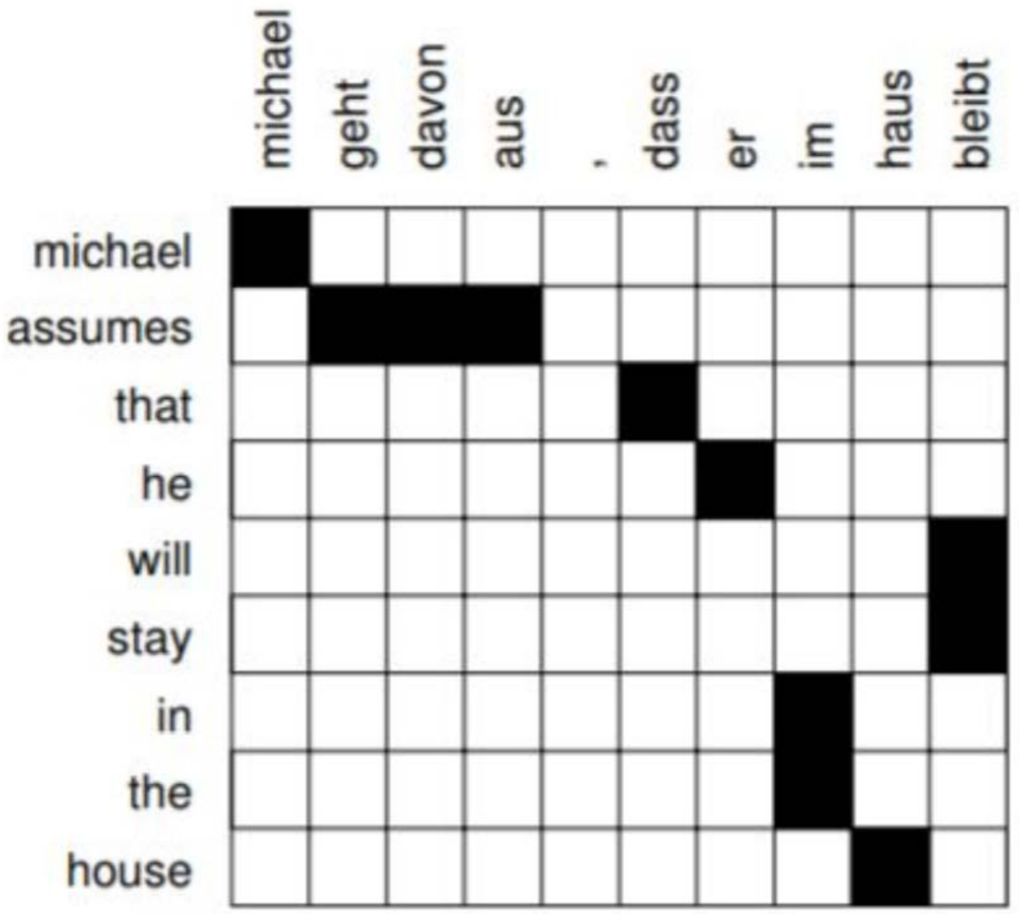
Convergence



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

Word Alignment:

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



Higher IBM Models

IBM Model 1	Lexical translation
Higher IBM Models IBM Model 2	Adds absolute reordering model
IBM Model 3	Adds fertility model
IBM Model 4	Relative reordering model
IBM Model 5	Fixes deficiency

- Only IBM Model 1 has global maximum
  - Training of a higher IBM model builds upon previous model
- Computationally biggest change in Model 3

IBM Model and EM

- IBM models create a many-to-one mapping
  - Words are aligned using an alignment function
  - A function may return the same value for different input (one-to-many mapping)
  - A function cannot return multiple values for one input (no many-to-one mapping)
- Real world alignments have many-to-many mappings

Decoding

- Goal is to find a translation that maximizes the product of the translation and language models.

$$\underset{E \in English}{\operatorname{argmax}} P(F \mid E) P(E)$$

- Cannot explicitly enumerate and test the combinatorial space of all possible translations.
- The optimal decoding problem for all reasonable model's (e.g. IBM model 1) is NP-complete.
- Heuristically search the space of translations using A*, beam-search, etc. to approximate the solution to this difficult optimization problem.

Evaluation Metrics

- Manual evaluation is most accurate, but expensive
- Automated evaluation metrics:
  - Compare system hypothesis with reference translations
  - BiLingual Evaluation Understudy (BLEU) (Papineni et al., 2002):
    - Modified n-gram precision

$$p_n = \frac{\text{number of } n\text{-grams appearing in both reference and hypothesis translations}}{\text{number of } n\text{-grams appearing in the hypothesis translation}}$$

BLEU

$$\text{BLEU} = \exp \frac{1}{N} \sum_{n=1}^{N} \log p_n$$

- Two modifications:
  - To avoid $\log 0$, all precisions are smoothed
  - Each n-gram in reference can be used at most once
    - Ex. **Hypothesis:** *to to to to to* vs **Reference:** *to be or not to be* should not get a unigram precision of 1
- Precision-based metrics favor short translations
  - Solution: Multiply score with a brevity penalty (BP) for translations shorter than reference, $e^{1-r/h}$

BLEU Scores

	Translation	p_1	p_2	p_3	p_4	BP	BLEU
Reference	*Vinay likes programming in Python*						
Sys1	*To Vinay it like to program Python*	$\frac{2}{7}$	0	0	0	1	.21
Sys2	*Vinay likes Python*	$\frac{3}{3}$	$\frac{1}{2}$	0	0	.51	.33
Sys3	*Vinay likes programming in his pajamas*	$\frac{4}{6}$	$\frac{3}{5}$	$\frac{2}{4}$	$\frac{1}{3}$	1	.76

8 8 3 3

BLEU

- Correlates somewhat well with human judgments

Problems with Lexical Translation

- Complexity – exponential in sentence length
- Weak reordering – the output is not fluent
- Many local decisions – error propagation