INF5820/INF9820

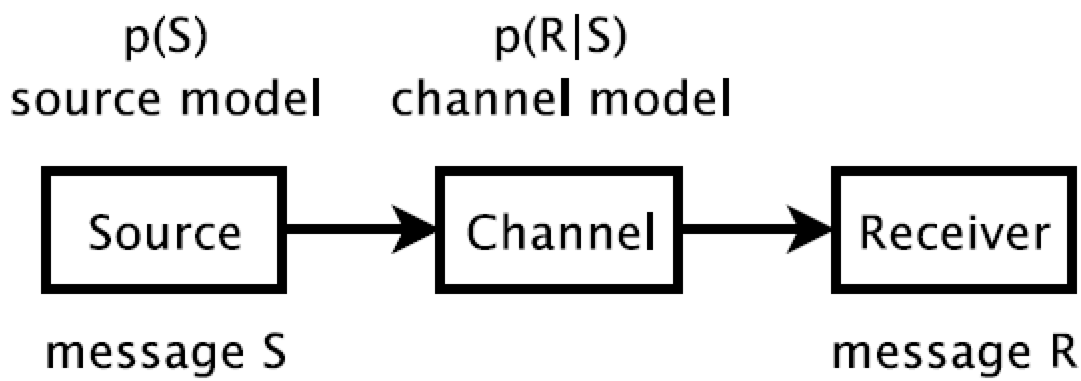
LANGUAGE TECHNOLOGICAL APPLICATIONS

Jan Tore Lønning, Lecture 4, 10 Sep.
jtl@ifi.uio.no

Today

- Statistical machine translation:
  - The noisy channel model
    - Word-based
      - IBM model 1
- Training

Noisy Channel Model



- Applying Bayes rule also called noisy channel model
  - we observe a distorted message R (here: a foreign string **f**)
  - we have a model on how the message is distorted (here: translation model)
  - we have a model on what messages are probably (here: language model)
  - we want to recover the original message S (here: an English string **e**)

SMT example

En	kokk	lagde	en	rett	med	bygg	.
a 0.9	chef 0.6	made 0.3	a 0.9	right 0.19	with 0.4	building 0.45	
…	cook 0.3	created 0.25	…	straight 0.17	by 0.3	construction 0.33	
	…	prepared 0.15		court 0.12	of 0.2	barley 0.11	
		constructed 0.12		dish 0.11	…	…	
		cooked 0.05		course 0.07			
		…		…			

Similarly for:
- pos 0-2 (2x3)
- pos 1-3
- pos 2-4
- pos 3-5 (4x5)
- pos 6-8

Pos4 – pos 6 (1x3x3 many)		Pos5 – pos 7 (5x3x3 many)	
a right with	2.7×10^{-12}	right with building	1.7×10^{-18}
a right of	1.5×10^{-10}	right with construction	5.4×10^{-18}
a right by	9.7×10^{-12}	right with barley	8.7×10^{-19}
…		…	
a course of	1.5×10^{-14}	course of barley	1.5×10^{-16}

Statistical Machine Translation - SMT

INF5820

Jan Tore Lønning

Department of Informatics
Universitety of Oslo

Statistical learning

Goal

- Find the best (most probable) English translation $\hat{E}$ of a foreign sentence F.
- $\hat{E} = \arg\max_{E} P(E \mid F)$

3 steps (common to many tasks)

1. A model. We may not have seen F before. The model will determine what to look for.
2. We must learn (or estimate) the parameters of the model from data.
3. We must have a method for using the model to find the best E given F, decoding.

Noisy channel models

- Applying Bayes' formula

$$\begin{aligned}\hat{E} &= \arg\max_E P(E \mid F) \\ &= \arg\max_E \frac{P(F \mid E)}{P(F)} P(E) \\ &= \arg\max_E P(F \mid E) P(E)\end{aligned}$$

- Turning the picture: consider F as a translation (distortion) of E, and ask which E?
- Why?
  - Suitable for approximations.
  - Makes use of language model $P(E)$.
- cf. K:SMT slide 34

Noisy channels

The noisy channel model

- See a distortion of the original.
- Goal: guess the original
- J&M Fig. 5.23, 9.2 og 25.15

Example

- Speech recognition: Sounds a distortion of writing.
- Tagging: Word sequence distortion of tag sequence
- Translation: Source language a distortion of target language.

Separating the models

Starting point:

$$\hat{E} = \arg\max_{E} P(F \mid E)P(E)$$

The models

- We can build and train two separate models:
  - The language model: $P(E)$
  - The translation model: $P(F \mid E)$
- Decoding must use both models simultaneously

Language model

Goal

Estimate the probability $P(E) = P(e_1 e_2 \dots e_n)$ of the string of words $e_1 e_2 \dots e_n$

n-gram model

$$
\begin{aligned}
& P(e_1 e_2 \dots e_n) \\
& \quad = \quad P(e_1)P(e_2 \mid e_1)P(e_3 \mid e_1, e_2) \cdots P(e_n \mid e_1 e_2 \dots e_{n-1}) \\
& \quad \approx \quad P(e_1)P(e_2 \mid e_1)P(e_3 \mid e_2) \cdots P(e_n \mid e_{n-1}) \\
& \quad = \quad P(e_1) \prod_{i=1}^{n-1} P(e_{i+1} \mid e_i)
\end{aligned}
$$

Comments:

- Uses the (incorrect) Markov-assumption
  $P(e_{(j+1)} \mid e_1 e_2 \ldots e_j) \approx P(e_{j+1} \mid e_j)$
- Last slide shows the bigram model. Could alternatively use trigram, quadgram, ...
- Trigram: $P(e_1 e_2 \ldots e_n) = \prod_{i=1}^{n-1} P(e_{i+1} \mid e_{i-1}, e_i)$
- For all n-grams : special symbols for start and end:
  - What is the probability of being the first word of a sentence?
  - What is the probability of being the last word of a sentence?

The translation model

Several alternatives:

- Word based
  - In particular the IBM-models: 1, 2, 3, 4, 5
- Phrase based
  - Parameter estimation often done on top of a word-based model.
- Syntax based

Word-based models

- Suppose
  - Source and target sentence always the same length
  - Word-order is preserved.
  - A one-to-one correspondence between words
- The translation would be like HMM-tagging

Translation	Tagging
source language word	word
target language word	tag
n-grams for targ. lang.	*n*-grams of tags
source sentence	sentence to be tagged
word translation probs.	probability for word given tag

- See simplified SMT example on slides from first MT lecture.

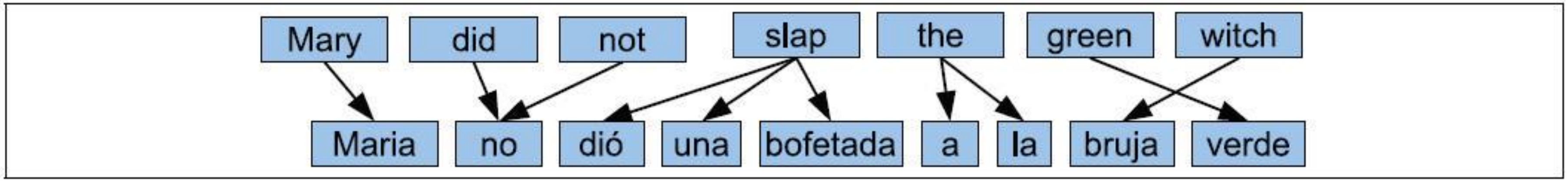
Word-based translation models

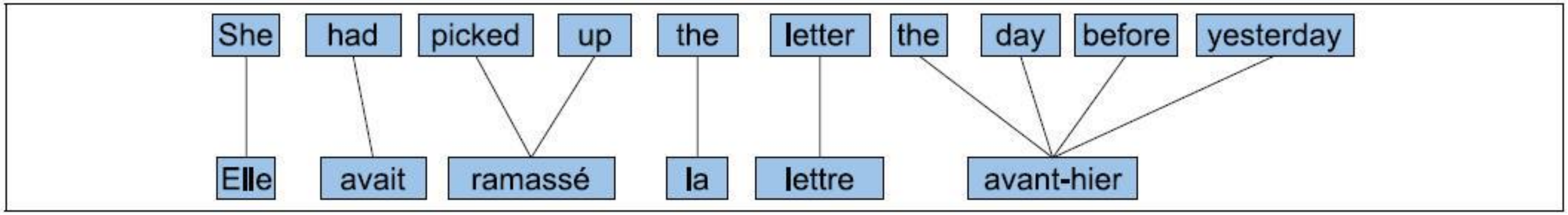
- But translation reorders, deletes, adds, goes many-to-one, one-to-many and many-to-many.
- We cannot apply HMM directly

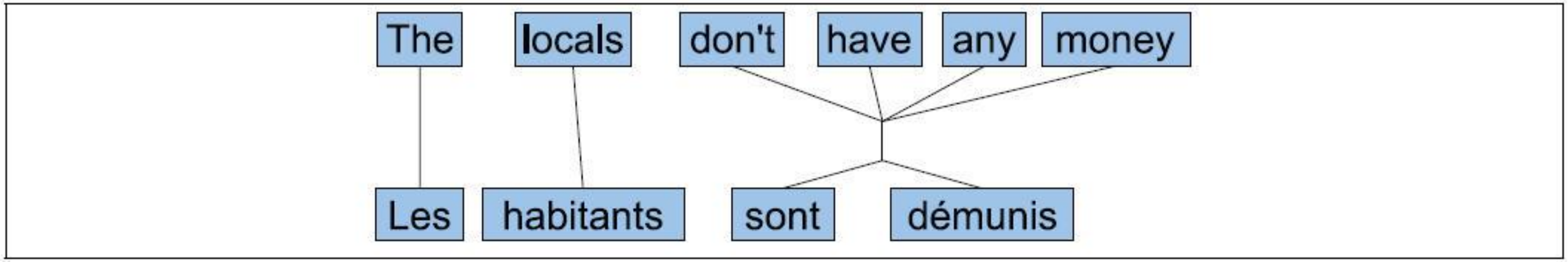
Two parts to word-based translation

1. What is the probability that source word *a* is translated as target word *b*?
2. Alignment: Which word(s) in the target language sentence is the translation of which word(s) in the source sentence?

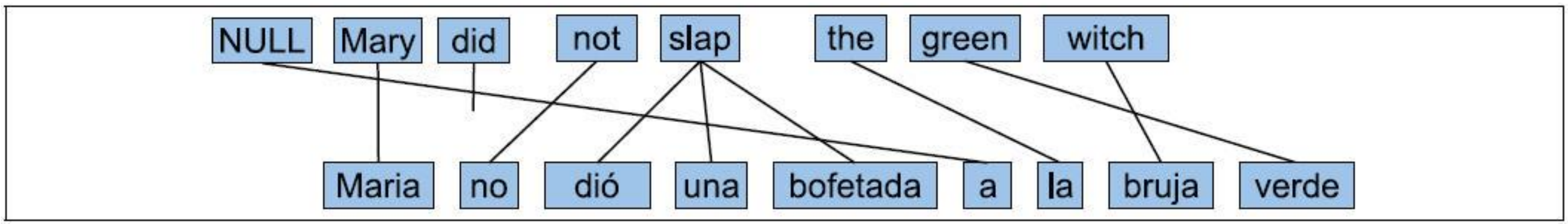
- J& M Figure 25.17, 25.20, 25.21, 25.22


Mary did not slap the green witch
Maria no dió una bofetada a la bruja verde


She had picked up the letter the day before yesterday
Elle avait ramassé la lettre avant-hier

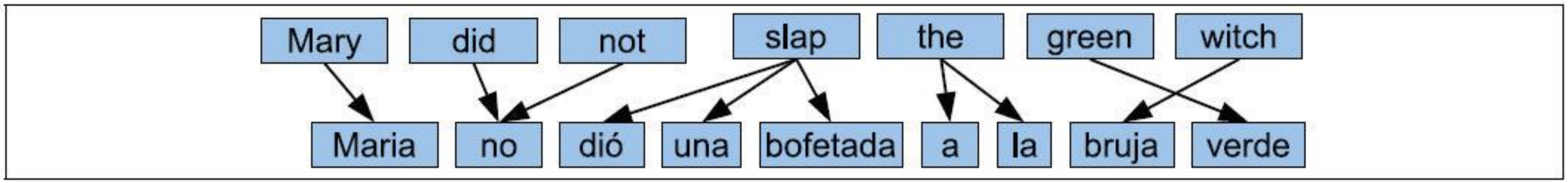

The locals don't have any money
Les habitants sont démunis

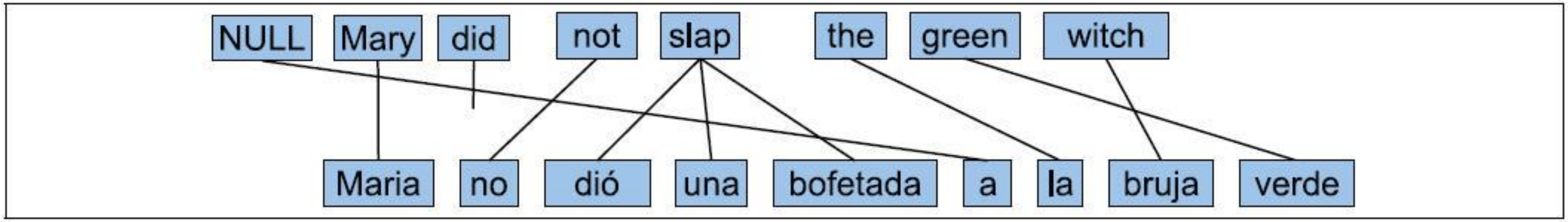
Alignment



- Length of English string: *k* (=7)
- Length of foreign string: *m* (=9)
- An alignment is a vector of length *m*, each entry a number between 0 and k
- The example:
  - $<a_1, a_2, \ldots, a_9> = <1, 3, 4, 4, 4, 0, 5, 7, 6>$

Alignment





- Artificial restrictions:
  - Several foreign words may be aligned with the same E word
  - A foreign word cannot be aligned to more than one E word

IBM Model 1

- Consider all possible alignments **a**:

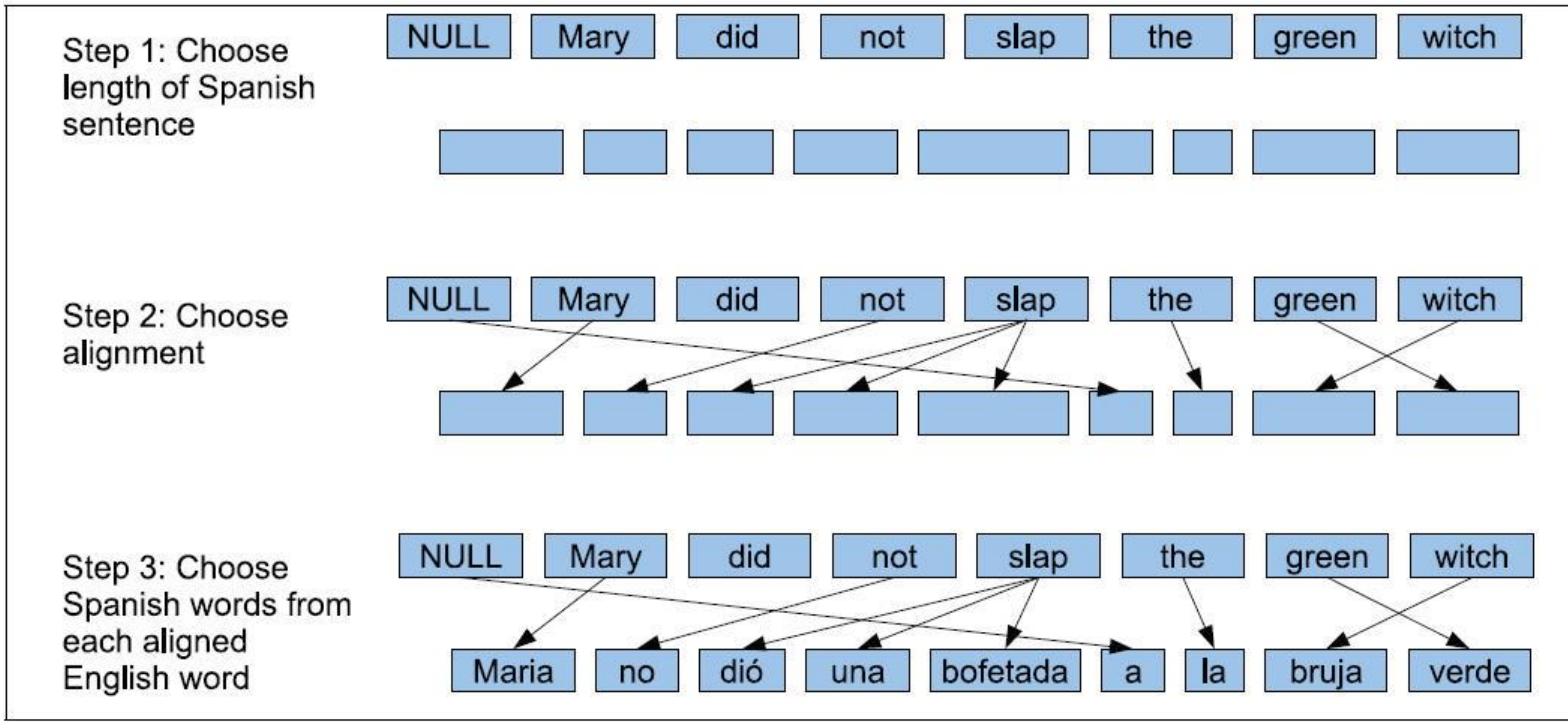
$$P(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$$

- For each alignment use the generative model:

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = P(m \mid \mathbf{e}) \prod_{j=1}^{m} P(a_j \mid a_1^{j-1}, f_1^{j-1}, m, \mathbf{e}) P(f_j \mid a_1^{j}, f_1^{j-1}, m, \mathbf{e})$$

- Simplify the model – make assumptions

Figure 25.23


Step 1: Choose length of Spanish sentence
NULL
Mary
did
not
slap
the
green
witch
Step 2: Choose alignment
NULL
Mary
did
not
slap
the
green
witch
Step 3: Choose Spanish words from each aligned English word
NULL
Mary
did
not
slap
the
green
witch
Maria
no
dió
una
bofetada
a
la
bruja
verde

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = P(m \mid \mathbf{e}) \prod_{j=1}^{m} P(a_j \mid a_1^{j-1}, f_1^{j-1}, m, \mathbf{e}) P(f_j \mid a_1^j, f_1^{j-1}, m, \mathbf{e})$$

- The generative model:
  - Choose the length of the foreign string $P(m \mid \mathbf{e})$
  - Which E word translates to the first F word $P(a_1 \mid m, \mathbf{e})$
  - What is the translation of this word? $P(f_1 \mid a_1, m, \mathbf{e})$
  - Which E word translates to the j-th F word given the choices so far $P(a_j \mid a_1^{j-1}, f_1^{j-1}, m, \mathbf{e})$
  - What is the translation of this word given the choices so far $P(f_j \mid a_1^j, f_1^{j-1}, m, \mathbf{e})$

Assumptions, approximations

$$P(\mathbf{f},\mathbf{a}\,|\,\mathbf{e}) = P(m\,|\,\mathbf{e})\prod_{j=1}^{m} P(a_j\,|\,a_1^{j-1}, f_1^{\,j-1}, m, \mathbf{e})P(f_j\,|\,a_1^{j}, f_1^{\,j-1}, m, \mathbf{e})$$

- $P(m\,|\,\mathbf{e})$ is a constant, independent of *m* and *E*
- $P(a_j\,|\,a_1^{j-1}, f_1^{\,j-1}, m, \mathbf{e}) = (k+1)^{-1}$
  - all alignments the same probability (adds to 1)
- $P(f_j\,|\,a_1^{j}, f_1^{\,j-1}, m, \mathbf{e}) = t(f_j\,|\,e_{a_j})$
  - the word translation probability only depends on source word

IBM model 1

$$P(\mathbf{f},\mathbf{a}\,|\,\mathbf{e}) = P(m\,|\,\mathbf{e})\prod_{j=1}^{m} P(a_j\,|\,a_1^{j-1}, f_1^{j-1}, m, \mathbf{e})P(f_j\,|\,a_1^{j}, f_1^{j-1}, m, \mathbf{e})$$

- Simplifies to

$$P(\mathbf{f},\mathbf{a}\,|\,\mathbf{e}) = \varepsilon\prod_{j=1}^{m}(k+1)^{-1}t(f_j\,|\,e_{a_j})$$

$$P(\mathbf{f},\mathbf{a}\,|\,\mathbf{e}) = \frac{\varepsilon}{(k+1)^m}\prod_{j=1}^{m}t(f_j\,|\,e_{a_j})$$

  - ε is a normalisation factor
  - Formula 4.7 in the SMT book
    - (The book goes f→ e, not e → f)

Parameter estimation

- If the training corpus was aligned, the model could be learned by counting:

$$t(f_j \mid e_{a_j}) = \frac{C(f_j, e_{a_j})}{\sum_f C(f, e_{a_j})}$$

- If we had known the translation probabilities, we could have found the most probable alignment.
- We neither know word probabilities nor alignment: Chicken and egg problem
- EM-algorithm: we may learn the two simultaneously

Training – the idea

1. From the translation probabilities, we may estimate alignment probabilities
   - (We do not choose only the best alignment)
2. From alignment probabilities, we may recalculate translation probabilities

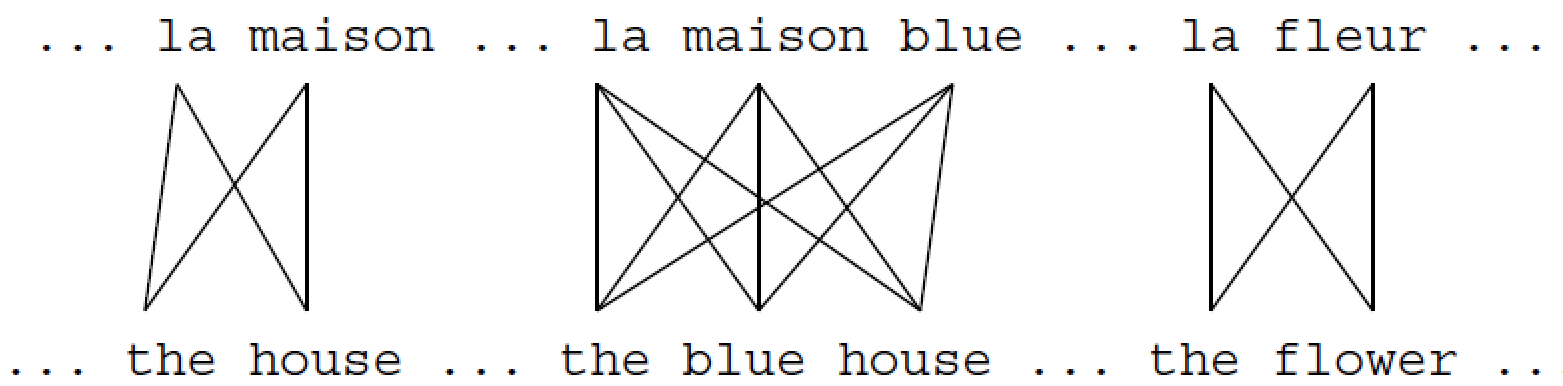
- By alternating between (1) and (2), the numbers converge towards better results
- For IBM Model 1 it may be proved that they converge towards a global optimum

EM Algorithm

- Incomplete data
  - if we had *complete data*, would could estimate *model*
  - if we had *model*, we could fill in the *gaps in the data*

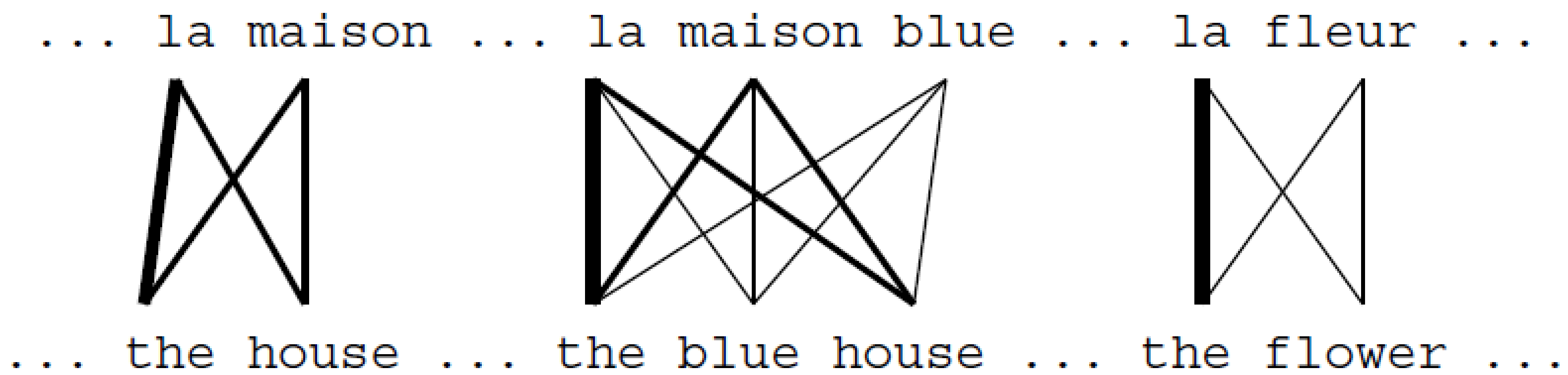
- Expectation Maximization (EM) in a nutshell
  1. initialize model parameters (e.g. uniform)
  2. assign probabilities to the missing data
  3. estimate model parameters from completed data
  4. iterate steps 2–3 until convergence

EM Algorithm



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

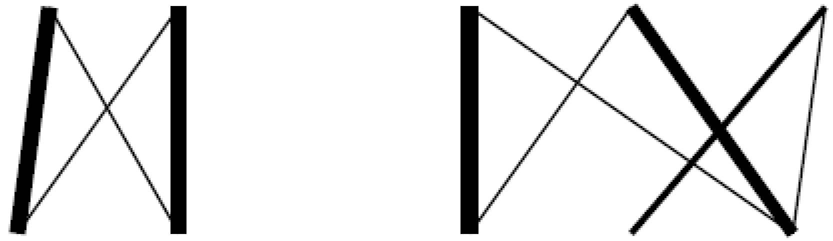
EM Algorithm



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

EM Algorithm

... la maison ... la maison bleu ... la fleur ...

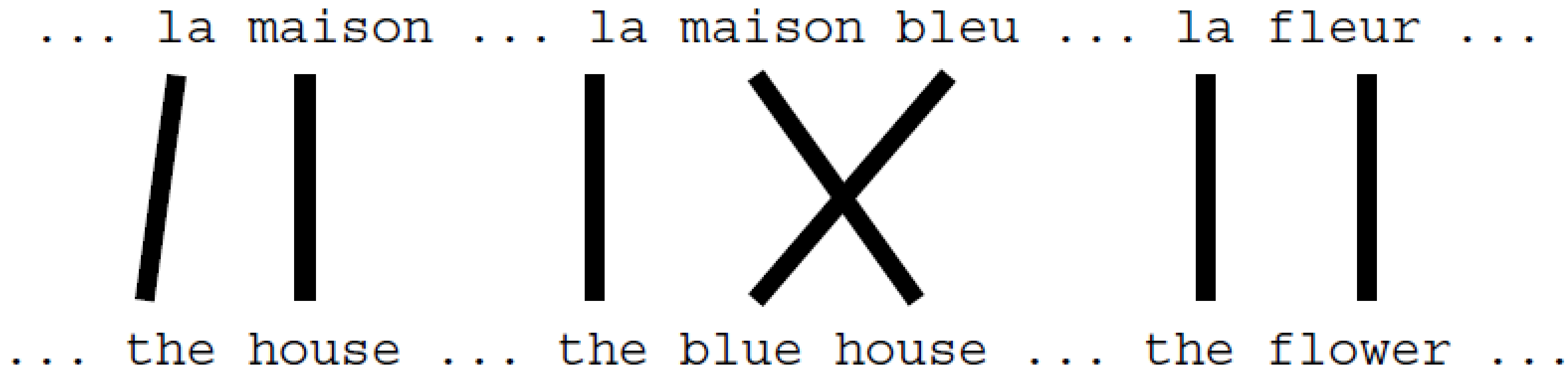


... the house ... the blue house ... the flower ...

- After another iteration

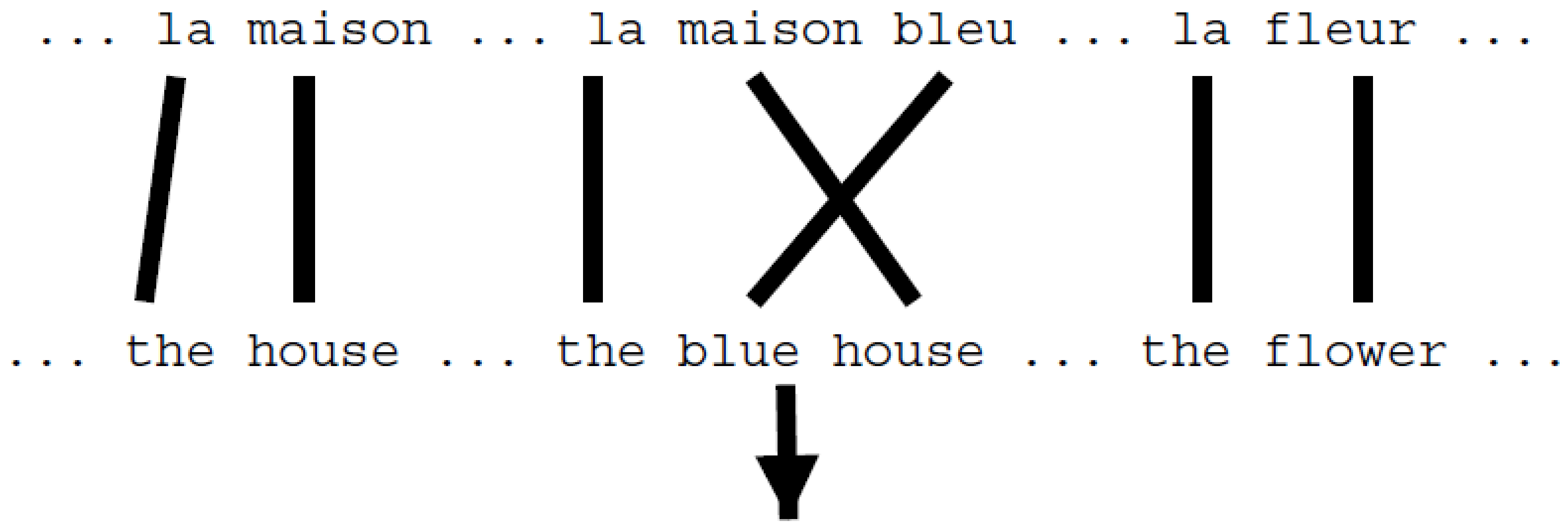
- It becomes apparent that alignments, e.g., between fleur and flower are mor likely (pigeon hole principle)

EM Algorithm



- Convergence
- Inherent hidden structure revealed by EM

EM Algorithm



$$\begin{aligned} \mathrm{p(la|the)} &= 0.453 \\ \mathrm{p(le|the)} &= 0.334 \\ \mathrm{p(maison|house)} &= 0.876 \\ \mathrm{p(bleu|blue)} &= 0.563 \\ &\ldots \end{aligned}$$

- Parameter estimation from the aligned corpus

Two ways to describe the algorithm

Intuitive

- Proceed
  - 1. Translation prob
  - 1. Alignment prob
  - 2. Translation prob
  - 2. Alignment prob
  - 3. Translation prob
  - Etc
- J&M, sec 25.6.1, example
- Intractable in practice

Efficient

- Sidestep alignment probs:
  - 1. Translation prob
  - 2. Translation prob
  - 3. Translation prob
  - Etc
- K:SMT, sec 4.2.3, example
- How it gets implemented

Training – the intuitive approach

1. Initalize the parameter values $t(f|e)$ for pairs of words f and e .
   - With no info, initalize them uniformly: Each word f in the foreign language is an equally likely translation of the word e.
2. For each pair $\boldsymbol{f}, \boldsymbol{e}$ of sentences in the corpus, use t to calculate the probabilities $P(\boldsymbol{a} \mid \boldsymbol{f}, \boldsymbol{e})$ to all possible alignments $\boldsymbol{a}$ of the two sentences.
   - (Called the expectation step, apply model to data)

Training – the intuitive approach

3. Collect fractional counts, $tc(f\,|e)$:

   («How many times e is translated as f»)

   1. First, calculate this, $c(f\,|e\,;\,\boldsymbol{f},\,\boldsymbol{e})$ for each sentence $\boldsymbol{f}$, $\boldsymbol{e}$, where we count:
      - how many times e is aligned to f by each alignment,
      - weighed by the probability of the alignment.
   2. Then add over all sentences to get

$$tc(f\,|\,e) = \sum_{(\mathbf{f},\mathbf{e})} c(f\,|\,e;\mathbf{f},\mathbf{e})$$

Training – the intuitive approach

4. Calculate the new translation probabilities

$$t(f|e) = \frac{tc(f|e)}{\sum_{f'} tc(f'|e)}$$

Errors in formula 4.14 in K:SMT

- where f' varies over all foreign words
- (Called the maximization step, estimate model from counts)

5. Repeat from 2 as long as you like

Assign probabilities to alignments

- Goal: compute $P(\mathbf{a} \mid \mathbf{f}, \mathbf{e})$
- Since
  $$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = P(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) P(\mathbf{f} \mid \mathbf{e})$$
  - we have
  $$P(\mathbf{a} \mid \mathbf{f}, \mathbf{e}) = \frac{P(\mathbf{f}, \mathbf{a} \mid \mathbf{e})}{P(\mathbf{f} \mid \mathbf{e})}$$
- We know
  $$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = \frac{\varepsilon}{(k+1)^m} \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

$$P(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$$

Example – the intuitive way

- Corpus

e_1: Dog barked
f_1: Hund bjeffet

e_2: Dog bit dog
f_2: Hund bet hund

3 English words: dog bit barked
3 foreign words: hund bjeffet bet

Step 1 initialization

t(hund|dog) = 1/3	t(bet|dog) = 1/3	t(bjeffet|dog) = 1/3
t(hund|bit) = 1/3	t(bet|bit) = 1/3	t(bjeffet|bit) = 1/3
t(hund|barked) = 1/3	t(bet|barked) = 1/3	t(bjeffet|barked) = 1/3
t(hund|0) = 1/3	t(bet|0) = 1/3	t(bjeffet|0) = 1/3

- Uniform
- Observe that we include the last line since an f-word may be aligned to 0.

Step 2: Alignment probabilities

e_1: Dog barked
f_1: Hund bjeffet

e_2: Dog bit dog
f_2: Hund bet hund

- Sentence pair 1:
  - 9 possible alignments:
    - <0,0>, <0,1>, <0,2>, <1,0>, <1,1>, <1,2>,<2,0>,<2,1>, <2,2>
  - Each equally probable: 1/9
  - (call this a_1: e.g. a_1(<0,1>)=1/27)
- Sentence pair 2:
  - 64 possible alignments:
    - <0,0,0>, <0,0,1>, … <3,3,3>
    - Each equally probable: 1/64
    - (call this a_2.)
    - Or, the hard way (next slide)

Step 2: The hard way

e$_2$: Dog bit dog
f$_2$: Hund bet hund

- Sentence pair 2:
  - 64 possible alignments:
    - <0,0,0>, <0,0,1>, … <3,3,3>
  - Each translation probability: 1/27

$$P(\mathbf{f}_2,<1,2,0>|\,\mathbf{e}_2)=\frac{\varepsilon}{(k+1)^m}\prod_{j=1}^{m}t(f_j\,|\,e_{a_j})=\frac{\varepsilon}{(3+1)^3}\prod_{j=1}^{3}t(f_j\,|\,e_{a_j})=\frac{\varepsilon}{4^3}t(f_1\,|\,e_1)\times t(f_2\,|\,e_2)\times t(f_3\,|\,e_0)=$$

$$\frac{\varepsilon}{4^3}t(hund\,|\,dog)\times t(bet\,|\,bit)\times t(hund\,|\,0)=\frac{\varepsilon}{4^3}\frac{1}{3}\times\frac{1}{3}\times\frac{1}{3}=\frac{\varepsilon}{4^3\times 3^3}$$

$$P(<1,2,0>|\,\mathbf{f}_2,\mathbf{e}_2)=\frac{P(\mathbf{f}_2,<1,2,0>|\,\mathbf{e}_2)}{P(\mathbf{f}_2\,|\,\mathbf{e}_2)}=\frac{P(\mathbf{f}_2,<1,2,0>|\,\mathbf{e}_2)}{\sum_{\mathbf{a}}P(\mathbf{a},\mathbf{f}_2\,|\,\mathbf{e}_2)}=\frac{\frac{\varepsilon}{64*27}}{64*\frac{\varepsilon}{64*27}}=\frac{1}{64}$$

Step 3.1: Collect fractional counts

Calculate $c(f \mid e\ ; \boldsymbol{f}, \boldsymbol{e})$ for each sentence $\boldsymbol{f}, \boldsymbol{e}$:

- Example: f =hund, e = dog, $\boldsymbol{f_1}, \boldsymbol{e_1}$:
  - There are 3 alignments that connect them: <1,0>, <1,1>, <1,2>
  - c(hund|dog; $\mathbf{f}_1$, $\mathbf{e}_1$) = a_1(<1,0>)+ a_1(<1,1>)+ a_1(<1,2>)=3*(1/9) = 1/3

$\mathbf{e}_1$: Dog barked
$\mathbf{f}_1$: Hund bjeffet

c(hund|dog; $\mathbf{f}_1$, $\mathbf{e}_1$)= 1/3	c(bjeffet|dog; $\mathbf{f}_1$, $\mathbf{e}_1$) = 1/3
c(hund|barked; $\mathbf{f}_1$, $\mathbf{e}_1$) = 1/3	c(bjeffet|barked; $\mathbf{f}_1$, $\mathbf{e}_1$) = 1/3
c(hund|0; $\mathbf{f}_1$, $\mathbf{e}_1$) = 1/3	c(bjeffet|0; $\mathbf{f}_1$, $\mathbf{e}_1$) = 1/3

Step 3.1: Collect frac. counts ctd

e_2: Dog bit dog
f_2: Hund bet hund

f_2, e_2:

- f =bet, e = bit
  - 16 alignments connect them: <x,2,z> for x,z in {0,1,2,3}
  - c(bet|bit; $\mathbf{f}_2$, $\mathbf{e}_2$) = 16/64 = 1/4
- f =bet, e = dog
  - all alignments <x,1,z> and <x,3,z> for x,z in {0,1,2,3}
  - c(bet|dog; $\mathbf{f}_2$, $\mathbf{e}_2$) = 2*16/64 = 1/2

c(hund|dog; $\mathbf{f}_2$, $\mathbf{e}_2$)= 1	c(bet|dog; $\mathbf{f}_2$, $\mathbf{e}_2$) = 1/2
c(hund|bit; $\mathbf{f}_2$, $\mathbf{e}_2$) = 1/2	c(bet|bit; $\mathbf{f}_2$, $\mathbf{e}_2$) = 1/4
c(hund|0; $\mathbf{f}_2$, $\mathbf{e}_2$) = 1/2	c(bet|0; $\mathbf{f}_2$, $\mathbf{e}_2$) = 1/4

Step 3.2: Total counts

$$tc(f \mid e) = \sum_{(\mathbf{f},\mathbf{e})} c(f \mid e; \mathbf{f}, \mathbf{e})$$

tc(hund|dog) = 1+1/3	tc(bet|dog) = 1/2	tc(bjeffet|dog) = 1/3	tc(*|dog)=4/3+1/2+1/3 =13/6
tc(hund|bit) = ½	tc(bet|bit) = ¼	tc(bjeffet|bit) = 0	tc(*|bit)=3/4
tc(hund|barked) = 1/3	tc(bet|barked) = 0	tc(bjeffet|barked) = 1/3	tc(*|barked) =2/3
tc(hund|0) = ½+1/3	tc(bet|0) = 1/4	tc(bjeffet|0) = 1/3	tc(*|0)=17/12

Step 4: new trans. probabilities

$$t(f|e) = \frac{tc(f|e)}{\sum_{f'} tc(f'|e)}$$

e	f	t(f|e)	exact	decimal
0	hund	(5/6)/(17/12)	10/17	0.588235
0	bet	(1/4)/(17/12)	3/17	0.176471
0	bjeffet	(1/3)/(17/12)	4/17	0.235294
dog	hund	(4/3)/(13/6)	8/13	0.615385
dog	bet	(1/2)/(13/6)	3/13	0.230769
dog	bjeffet	(1/3)/(13/6)	2/13	0.153846
bit	hund	(1/2)/(3/4)	2/3	0.666667
bit	bet	(1/4)/(3/4)	1/3	0.333333
barked	hund	(1/3)/(2/3	1/2	0.5
barked	bjeffet	(1/3)/(2/3)	1/2	0.5

Repeat: Step 2, sentence 1

- 9 different alignments
- $P'(\mathbf{a}) = c\ P(\mathbf{a}, \mathbf{f_1} \mid \mathbf{e_1})$
- $P(\mathbf{a}) = P(\mathbf{a} \mid \mathbf{e_1}, \mathbf{f_1})$

$\mathbf{e_1}$: Dog barked
$\mathbf{f_1}$: Hund bjeffet

			P’	P=P’/1,4145436
P’(<0,0>) =	t(hund|0)*t(bjeffet|0)=	(10/17)*(3/17)=	0,103806	0,0733848
P’(<0,1>)=	t(hund|0)*t(bjeffet|dog)=	(10/17)*(2/13)=	0,0904977	0,0639766
P’(<0,2>)=	t(hund|0)*t(bjeffet|barked)=	(10/17)*(1/2)=	0,294118	0,207924
P’(<1,0>) =	t(hund|dog)*t(bjeffet|0)=	(8/13)*(3/17)=	0,108597	0,0767718
P’(<1,1>) =	t(hund|dog)*t(bjeffet|dog)=	(8/13)*(2/13)=	0,0946746	0,0669294
P’(<1,2>) =	t(hund|dog)*t(bjeffet|barked)=	(8/13)*(1/2)=	0,307692	0,217520
P’(<2,0>) =	t(hund|barked)*t(bjeffet|0)=	(1/2)*(3/17)=	0,0882352	0,06237715
P’(<2,1>)=	t(hund|barked)*t(bjeffet|dog)=	(1/2)*(2/13)=	0,0769231	0,05438015
P’(<2,2>)=	t(hund|barked)*t(bjeffet|barked)=	(1/2)*(1/2)=	0,25	0,176735
Sum of P’s			1,4145436	

Repeat: Step 2, sentence 2

- 64 different alignments
- Home work til next week!
- How many alignments if the sentences are 10 words long?
- That's why we need a smarter way.
- To be continued …