Machine Translation
Word-based models and the EM algorithm

Miles Osborne
(Original slides by Philipp Koehn and Barry Haddow)

January 2013

Lexical translation

- How to translate a word → look up in dictionary

  Haus — *house, building, home, household, shell.*

- *Multiple translations*
  - some more frequent than others
  - for instance: *house*, and *building* most common
  - special cases: *Haus* of a *snail* is its *shell*

- Note: During all the lectures, we will translate from a foreign language into English

Collect statistics

- Look at a *parallel corpus* (German text along with English translation)

Translation of $Haus$	Count
house	8,000
building	1,600
home	200
household	150
shell	50

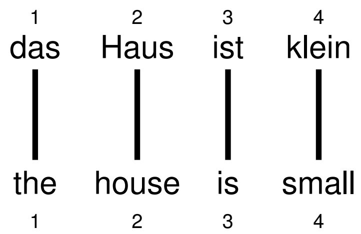
Estimate translation probabilities

- *Maximum likelihood estimation*

$$p_f(e) = \begin{cases} 0.8 & \text{if } e = \textit{house}, \\ 0.16 & \text{if } e = \textit{building}, \\ 0.02 & \text{if } e = \textit{home}, \\ 0.015 & \text{if } e = \textit{household}, \\ 0.005 & \text{if } e = \textit{shell}. \end{cases}$$

Alignment

- In a parallel text (or when we translate), we **align** words in one language with the words in the other



- Word *positions* are numbered 1–4

Alignment

Alignments:

- Specify word translations
- Allow us to recover word order

They are not given, so we need *EM*

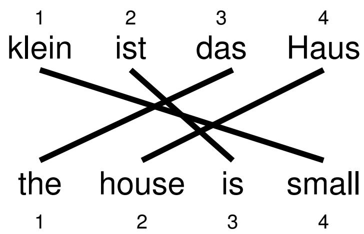
Alignment function

- Formalizing *alignment* with an **alignment function**
- Mapping an English target word at position i to a German source word at position j with a function $a : i \to j$
- Example

$$a : \{1 \to 1, 2 \to 2, 3 \to 3, 4 \to 4\}$$

Reordering

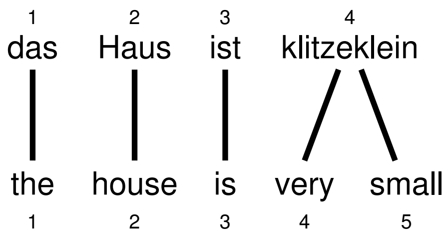
- Words may be **reordered** during translation



$$a : \{1 \to 3, 2 \to 4, 3 \to 2, 4 \to 1\}$$

One-to-many translation

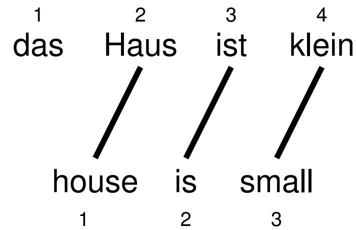
- A source word may translate into **multiple** target words



$$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4, 5 \rightarrow 4\}$$

Dropping words

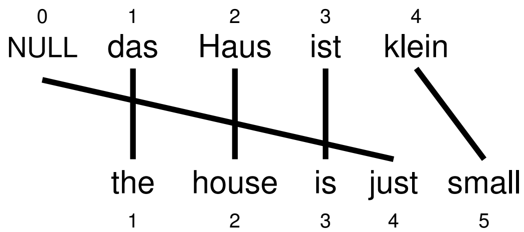
- Words may be **dropped** when translated
  - The German article *das* is dropped



$$a : \{1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4\}$$

Inserting words

- Words may be **added** during translation
  - The English *just* does not have an equivalent in German
  - We still need to map it to something: special NULL token



$$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 0, 5 \rightarrow 4\}$$

IBM Models

- SMT systems are based upon aligned parallel corpora.

- We use machine learning to tell us this alignment.

IBM Models are a series of translation models for word alignment

IBM Model 1

IBM Model 1 only uses *lexical translation*

- Translation probability
  - for a foreign sentence **f** $= (f_1, ..., f_{l_f})$ of length l_f
  - to an English sentence **e** $= (e_1, ..., e_{l_e})$ of length l_e
  - with an alignment of each English word e_j to a foreign word $f_{a(j)}$ according to the alignment function $a : j \rightarrow i$

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

  - parameter ϵ is a *normalisation constant*

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

$$p(e, a|f) = \frac{\epsilon}{5^4} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein})$$
$$= \frac{\epsilon}{5^4} \times 0.7 \times 0.8 \times 0.8 \times 0.4$$

Learning lexical translation models

- We would like to *estimate* the lexical translation probabilities $t(e|f)$ from a parallel corpus

- ... but we do not have the alignments

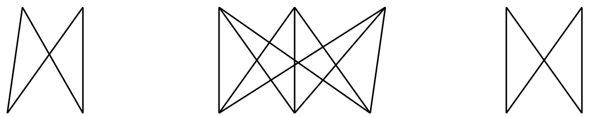
- **Chicken and egg problem**
  - if we had the *alignments*,
    $\rightarrow$ we could estimate the *parameters* of our generative model
  - if we had the *parameters*,
    $\rightarrow$ we could estimate the *alignments*

EM algorithm

- **Expectation Maximization (EM)** in a nutshell
  - initialise model parameters (e.g. uniform)
  - assign probabilities to the missing data (ie guess probability of data given model)
  - (re) estimate model parameters from completed data
  - iterate

EM algorithm

`... la maison ... la maison blue ... la fleur ...`

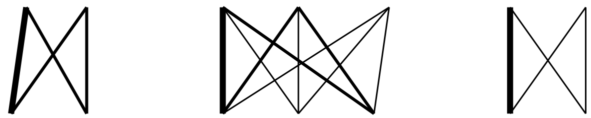


`... the house ... the blue house ... the flower ...`

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

EM algorithm

`... la maison ... la maison blue ... la fleur ...`

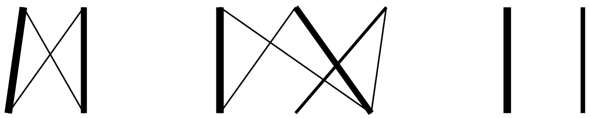


`... the house ... the blue house ... the flower ...`

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

EM algorithm

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

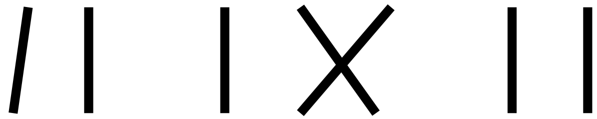


... the house ... the blue house ... the flower ...

- After another iteration
- It becomes apparent that alignments, e.g., between *fleur* and *flower* are more likely

EM algorithm

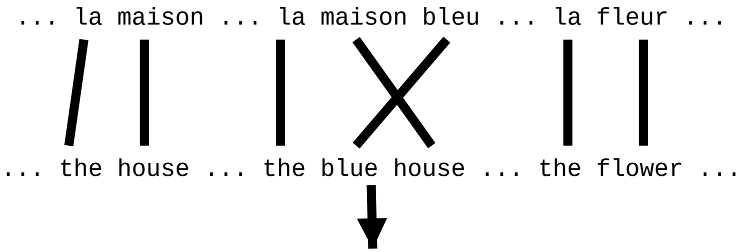
... la maison ... la maison bleu ... la fleur ...



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

- Convergence
- Inherent hidden structure revealed by EM

EM algorithm




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

- Parameter estimation from the aligned corpus

IBM Model 1 and EM

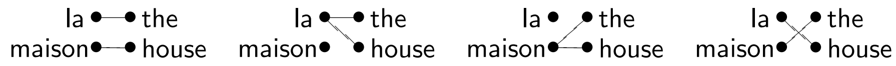
- EM Algorithm consists of two steps
- **Expectation-Step**: Apply model to the data
  - parts of the model are hidden (here: alignments)
  - using the model, assign probabilities to possible values
- **Maximization-Step**: Estimate model from data
  - take assigned values as fact
  - collect counts (weighted by probabilities)
  - estimate model from counts
- Iterate these steps until **convergence**

IBM Model 1 and EM

- We need to be able to:
  - Compute probability of alignments (for weighing counts)
  - Collect counts and update parameters

IBM Model 1 and EM

- **Probabilities** $p(\text{the}|\text{la}) = 0.7$ $p(\text{house}|\text{la}) = 0.05$ $p(\text{the}|\text{maison}) = 0.1$ $p(\text{house}|\text{maison}) = 0.8$
- **Alignments**



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

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

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

IBM Model 1 and EM: Expectation Step

- We need to compute $p(a|\mathbf{e}, \mathbf{f})$
- Applying the *chain rule*:

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

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

IBM Model 1 and EM: Expectation Step

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

$$p(\mathbf{e}|\mathbf{f}) = \sum_a p(\mathbf{e}, a|\mathbf{f})$$
$$= \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)$$

- This sums over a possibly exponential number of alignments
- Algebraic manipulation reduces the computation and makes it tractable

IBM Model 1 and EM: Expectation Step

- Combine what we have:

$$
\begin{aligned}
p(a|\mathbf{e},\mathbf{f}) &= p(\mathbf{e},a|\mathbf{f})/p(\mathbf{e}|\mathbf{f}) \\
&= \frac{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})}{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)} \\
&= \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)}
\end{aligned}
$$

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:

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

Note:

- $\delta(e,e_j) = 1$ if word e appears in position e_j, 0 otherwise

- These are expected (not true or observed) counts

IBM Model 1 and EM: Maximization Step

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

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

(ie for a given word f, count how many times it was translated to a given e and normalise by the number of times f was translated to any word)

IBM Model 1 and EM: Pseudocode

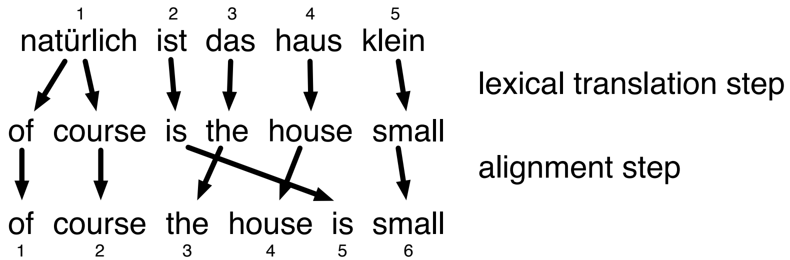
```
initialize t(e|f) uniformly
do until convergence
  set count(e|f) to 0 for all e,f
  set total(f) to 0 for all f
  for all sentence pairs (e_s,f_s)
    for all words e in e_s
      total_s(e) = 0
      for all words f in f_s
        total_s(e) += t(e|f)
    for all words e in e_s
      for all words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

Higher IBM Models

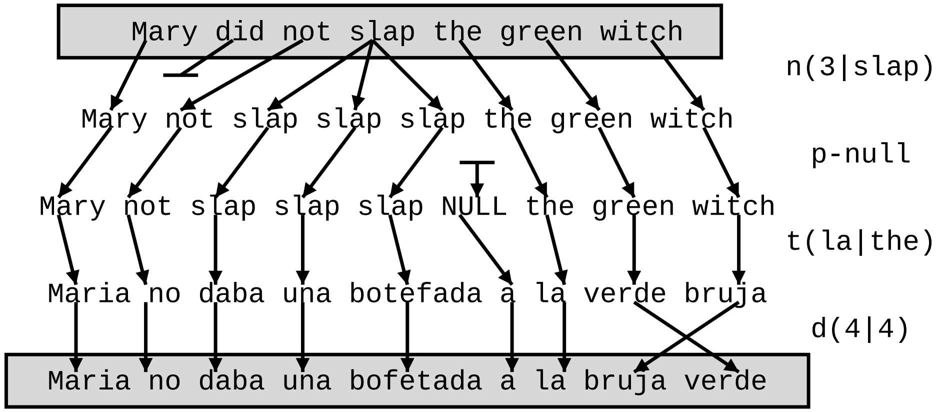
IBM Model 1	lexical translation
IBM Model 2	adds absolute **reordering model**
IBM Model 3	adds **fertility model**
IBM Model 4	relative reordering model
IBM Model 5	fixes **deficiency**

- Only IBM Model 1 has *global maximum*
  - training of a higher IBM model builds on previous model
- Computationally biggest change in Model 3
  - trick to simplify estimation does not work anymore
  - → *exhaustive* count collection becomes computationally too expensive
  - **sampling** over high probability alignments is used instead

IBM Model 2



IBM Model 3



Summary

- Word-based Models
  - Lexical translation
  - Alignment
- IBM Model 1
- Training with EM
- Higher IBM Models