Word-based models

Dr. Mariana Neves
(adapted from the original slides
of Prof. Philipp Koehn)

November 7th, 2016

Overview

- IBM Model 1
- IBM Model 2
- IBM Model 3
- IBM Model 4
- IBM Model 5

- This model generates many different translations for a sentence, each with a different probability
- The estimation is based on the individual words, not on the whole sentence

Generative modeling

- breaks up the process in many smaller steps,
- models these steps with probability distributions,
- and combines the steps into a coherent story

IBM Model 1

- IBM Model 1 only uses lexical translation
- Translation probability
  - for a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$ of length l_f
  - to an English sentence $\mathbf{e} = (e_1, ..., e_{l_e})$ of length l_e
  - with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \to i$

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

IBM Model 1

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

- The right side is the product over the **lexical translation probabilities** for all l_e generated output words e_j.

IBM Model 1

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

- The left side is a fraction necessary for normalization.
- It uses $(l_f + 1)$ input tokens because we also consider the NULL token.
- There are $(l_f + 1)^{l_e}$ different alignments that map $(l_f + 1)$ input words into l_e output words.
- parameter ϵ is a normalization 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

$$\begin{aligned} p(e,a|f) &= \frac{\epsilon}{5^4} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein}) \\ &= \frac{\epsilon}{5^4} \times 0.7 \times 0.8 \times 0.8 \times 0.4 \\ &= 0.0028\epsilon \end{aligned}$$

Learning the translation probability distributions

- We will learn these probabilities based on sentence-aligned paired texts
- Corpora are not usually word-aligned, just sentence-aligned
- Problem of incomplete data
- Typical problem in machine learning which is usually modeled as a hidden variable

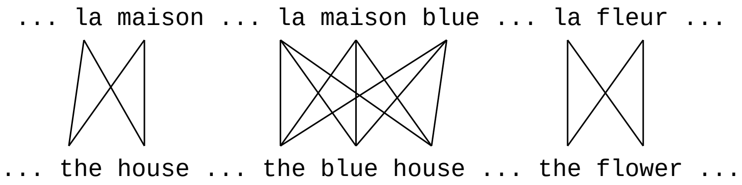
Learning Lexical Translation Models

- 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

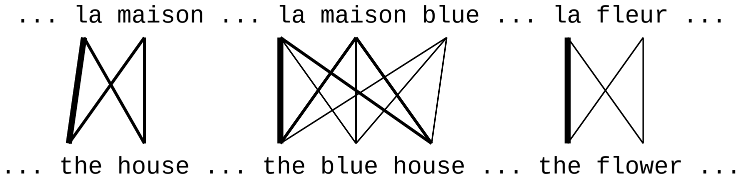
- Incomplete data
  - if we had *complete data*, would could estimate *model*
  - if we had *model*, we could fill in the *gaps in the data*
- Expectation Maximization (EM) in a nutshell
  1. initialize model parameters (e.g. uniform)
  2. assign probabilities to the missing data
  3. estimate model parameters from completed data
  4. iterate steps 2–3 until convergence

EM Algorithm

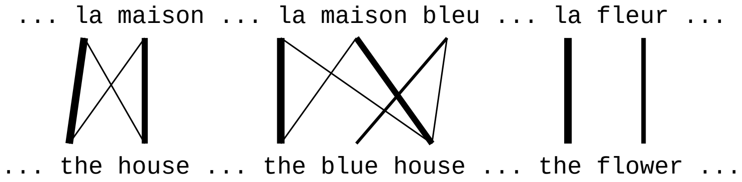


- Initial step: all alignments equally likely
- Model learns that, e.g., la is often aligned with the

EM Algorithm

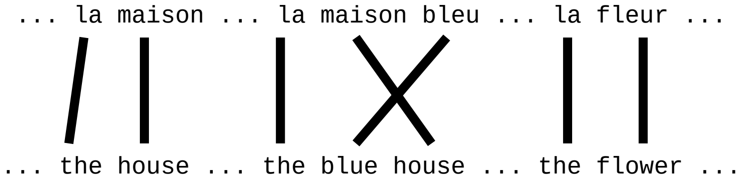


- After one iteration
- Alignments, e.g., between la and the are more likely



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

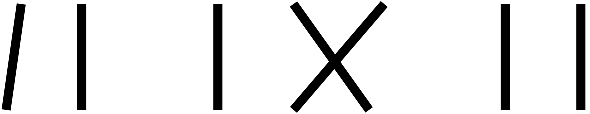
EM Algorithm



- Convergence
- Inherent hidden structure revealed by EM

EM Algorithm

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



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

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 assign 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:
  - Expectation-Step: probability of alignments
  - Maximization-Step: count collection

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 $p(\mathbf{e}, \mathbf{a}|\mathbf{f})$ (definition of Model 1)

IBM Model 1 and EM: Expectation Step

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

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

IBM Model 1 and EM: Expectation Step

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

$$= \frac{\epsilon}{(l_f+1)^{l_e}} \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

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

- Note the trick in the last line
  - removes the need for an exponential number of products
  - → this makes IBM Model 1 estimation tractable

The Trick

(case $l_e = l_f = 2$)

$$\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} = \frac{\epsilon}{3^2} \prod_{j=1}^{2} t(e_j|f_{a(j)}) =$$

$$= t(e_1|f_0)\ t(e_2|f_0) + t(e_1|f_0)\ t(e_2|f_1) + t(e_1|f_0)\ t(e_2|f_2) +$$
$$+ t(e_1|f_1)\ t(e_2|f_0) + t(e_1|f_1)\ t(e_2|f_1) + t(e_1|f_1)\ t(e_2|f_2) +$$
$$+ t(e_1|f_2)\ t(e_2|f_0) + t(e_1|f_2)\ t(e_2|f_1) + t(e_1|f_2)\ t(e_2|f_2) =$$
$$= t(e_1|f_0)\,(t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) +$$
$$+ t(e_1|f_1)\,(t(e_2|f_1) + t(e_2|f_1) + t(e_2|f_2)) +$$
$$+ t(e_1|f_2)\,(t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2)) =$$
$$= (t(e_1|f_0) + t(e_1|f_1) + t(e_1|f_2))\,(t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2))$$

IBM Model 1 and EM: Expectation Step

- Combine what we have:

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

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

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

IBM Model 1 and EM: Maximization Step

- Now we have to collect counts over all possible alignments, weighted by their probabilities
- Evidence from a sentence pair **e,f** that word e is a translation of word f:

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

- With the same simplification as before:

$$c(e|f;\mathbf{e},\mathbf{f}) = \frac{t(e|f)}{\sum_{i=0}^{l_f} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e,e_j) \sum_{i=0}^{l_f} \delta(f,f_i)$$

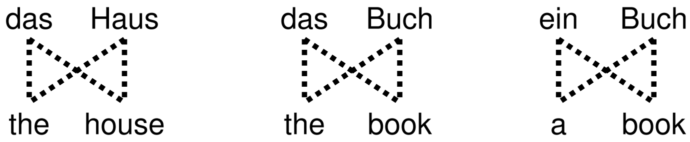
IBM Model 1 and EM: Maximization Step

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

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

IBM Model 1 and EM: Pseudo-code

```
Input: set of sentence pairs (e, f)
Output: translation prob. t(e|f)
initialize t(e|f) uniformly
while not converged do
    // initialize
    count(e|f) = 0 for all e, f
    total(f) = 0 for all f
    for all sentence pairs (e,f) do
        // compute normalization
        for all words e in e do
            s-total(e) = 0
            for all words f in f do
                s-total(e) += t(e|f)
            end for
        end for
        // collect counts
        for all words e in e do
            for all words f in f do
                count(e|f) += t(e|f) / s-total(e)
                total(f) += t(e|f) / s-total(e)
            end for
        end for
    end for
    // estimate probabilities
    for all foreign words f do
        for all English words e do
            t(e|f) = count(e|f) / total(f)
        end for
    end for
end while
```



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

Convergence of EM

- How can we measure whether our model converged?
- We are building a model for translation and we want it to perform well when translating unseen sentences.

Convergence of EM

Starting with the uniform probabilities:

$$p(\textit{the book}|\textit{das Buch}) = \frac{\epsilon}{2^2}(0.25 + 0.25)(0.25 + 0.25) = 0.0625\epsilon$$

Convergence of EM

After the first iteration:

$$t(e|f) = \begin{cases} 0.5 & \text{if } e = \text{the and } f = \text{das} \\ 0.25 & \text{if } e = \text{the and } f = \text{buch} \\ 0.25 & \text{if } e = \text{book and } f = \text{das} \\ 0.5 & \text{if } e = \text{book and } f = \text{buch} \end{cases}$$

$$p(the\ book|das\ Buch) = \frac{\epsilon}{2^2}(0.5 + 0.25)(0.25 + 0.5) = 0.140625\epsilon$$

Convergence of EM

This will ultimately converge to:

$$t(e|f) = \begin{cases} 1 & \text{if } e = \text{the and } f = \text{das} \\ 0 & \text{if } e = \text{the and } f = \text{buch} \\ 0 & \text{if } e = \text{book and } f = \text{das} \\ 1 & \text{if } e = \text{book and } f = \text{buch} \end{cases}$$

$$p(\textit{the book}|\textit{das Buch}) = \frac{\epsilon}{2^2}(1+0)(0+1) = 0.25\epsilon$$

Perplexity

- How well does the model fit the data?
- Perplexity: derived from probability of the training data according to the model

$$\log_2 PP = -\sum_s \log_2 p(\mathbf{e}_s|\mathbf{f}_s)$$

Perplexity

- Example (ϵ=1)

	initial	1st it.	2nd it.	3rd it.	...	final
$p(\text{the haus}\|\text{das haus})$	0.0625	0.1875	0.1905	0.1913	...	0.1875
$p(\text{the book}\|\text{das buch})$	0.0625	0.1406	0.1790	0.2075	...	0.25
$p(\text{a book}\|\text{ein buch})$	0.0625	0.1875	0.1907	0.1913	...	0.1875
perplexity	4095	202.3	153.6	131.6	...	113.8

- The perplexity is guaranteed to decrease or stay the same in each iteration.
- In the IBM model 1, the EM training will eventually reach a global minimum.

Ensuring Fluent Output

- Our translation model cannot decide between small and little
- Sometime one is preferred over the other:
  - small step: 2,070,000 occurrences in the Google index
  - little step: 257,000 occurrences in the Google index
- Language model
  - estimate how likely a string is English
  - based on n-gram statistics
    - unigram: when considering a single word (e.g., small)
    - bigram: when considering a sequence of two consecutive words (e.g., small step)
    - trigram: when considering a sequence of three consecutive words (e.g., small step to)

N-gram Language Models

- We break the long sentences into smaller steps for which we can collect sufficient statistics.
- For instance, trigram models (n=3):

$$
\begin{aligned}
p(\mathbf{e}) &= p(e_1, e_2, ..., e_n) \\
&= p(e_1)p(e_2|e_1)...p(e_n|e_1, e_2, ..., e_{n-1}) \\
&\simeq p(e_1)p(e_2|e_1)...p(e_n|e_{n-2}, e_{n-1})
\end{aligned}
$$

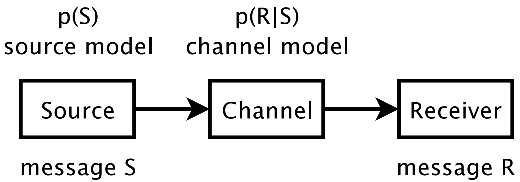
- Statistics can be computed based on both the English dataset of the parallel corpus.
- But also on any text resource in this language (English).

Noisy Channel Model

- We would like to integrate a language model.
- We look for the best translation e for the input foreign sentence f.
- Use use Bayes rule to include $p(e)$:

$$\begin{aligned} \text{argmax}_{\mathbf{e}}\ p(\mathbf{e}|\mathbf{f}) &= \text{argmax}_{\mathbf{e}} \frac{p(\mathbf{f}|\mathbf{e})\ p(\mathbf{e})}{p(\mathbf{f})} \\ &= \text{argmax}_{\mathbf{e}}\ p(\mathbf{f}|\mathbf{e})\ p(\mathbf{e}) \end{aligned}$$

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**)

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

Reminder: IBM Model 1

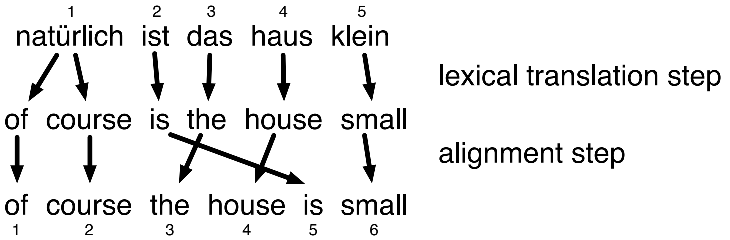
Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

- Generative model: break up translation process into smaller steps
  - IBM Model 1 only uses lexical translation
- Translation probability
  - for a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$ of length l_f
  - to an English sentence $\mathbf{e} = (e_1, ..., e_{l_e})$ of length l_e
  - with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \to i$

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

  - parameter ϵ is a normalization constant

Adding a model of alignment:

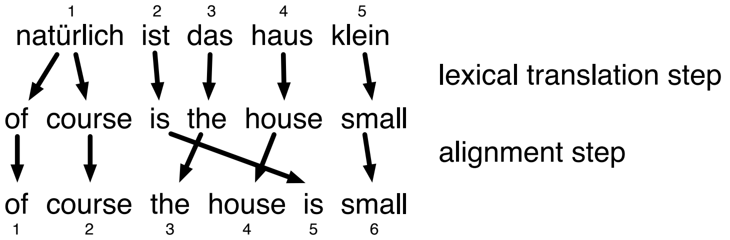


Alignment probability

- We model alignment with an **alignment probability distribution**.
- We translate foreign word at position i to English word at position j:

$$a(i|j, l_e, l_f)$$

IBM Model 2 - Example



We have a two-step process:

- lexical translation step: translation probability (e.g., $t(is|ist)$)
- alignment step: alignment probability (e.g., $a(2|5,6,5)$)

- Putting everything together

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

- EM training of this model works the same way as IBM Model 1

IBM Model 2: Expectation Step

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

- We use the same trick, just like Model 1

IBM Model 2: Maximization Step

- We can compute the fractional counts for **lexical translations**:

$$c(e|f;\mathbf{e},\mathbf{f}) = \sum_{j=1}^{l_e}\sum_{i=0}^{l_f} \frac{t(e|f)a(i|j,l_e,l_f)\delta(e,e_j)\delta(f,f_i)}{\sum_{i'=0}^{l_f} t(e|f_{i'})a(i'|j,l_e,l_f)}$$

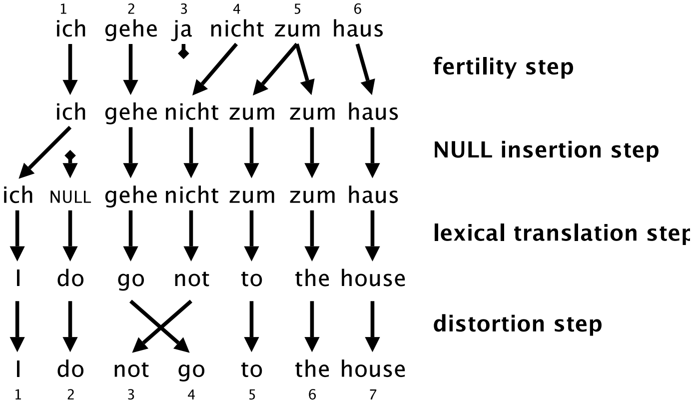
- and the counts for **alignments**:

$$c(i|j;l_e,l_f\mathbf{e},\mathbf{f}) = \frac{t(e_j|f_i)a(i|j,l_e,l_f)}{\sum_{i'=0}^{l_f} t(e|f_{i'})a(i'|j,l_e,l_f)}$$

IBM Model 2

- It is very similar to that for IBM Model 1.
- But we do not initialize the probabilities for $t(e|f)$ and $a(i|j, l_e, l_f)$ uniformly.
  - We get estimations from a few iterations of Model 1 instead.
  - Model 1 is a special case of Model 2 with $a(i|j, l_e, l_f)$ fixed to $\frac{1}{l_f+1}$.

IBM Model 3


1 2 3 4 5 6
ich gehe ja nicht zum haus
ich gehe nicht zum zum haus
ich NULL gehe nicht zum zum haus
I do go not to the house
I do not go to the house
1 2 3 4 5 6 7
fertility step
NULL insertion step
lexical translation step
distortion step

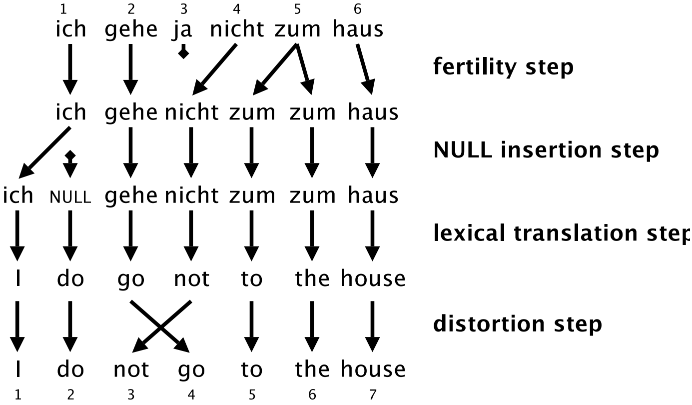
Fertility

- Fertility: number of English words generated by a foreign word
- Modeled by distribution $n(\phi|f)$, in which $\phi = 0, 1, 2, ...$
- Example:

$$n(1|haus) \simeq 1$$
$$n(2|zum) \simeq 1$$
$$n(0|ja) \simeq 1$$

Fertility - NULL token

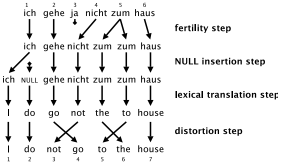
- Modeled by distribution $n(\phi|NULL)$
- This is modeled as a special step as inserted words depends on the sentence length.
  - probability p_1 to introduce a NULL token
  - or probability $p_0 = 1 - p_1$ **not** to introduce a NULL token


1 2 3 4 5 6
ich gehe ja nicht zum haus
fertility step
ich gehe nicht zum zum haus
NULL insertion step
ich NULL gehe nicht zum zum haus
lexical translation step
I do go not to the house
distortion step
I do not go to the house
1 2 3 4 5 6 7

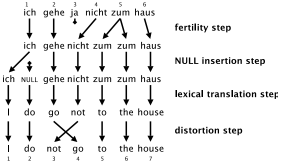
IBM Model 3 - four-step process

- Fertility: modeled by $n(\phi|f)$, e.g., $n(2|zum)$.
- NULL insertion: modeled by p_1 (e.g., NULL insertion after ich), and $p_0 = 1 - p_1$ (e.g., no NULL insertion after nicht).
- Lexical translation: modeled by $t(e|f)$ (Model 1), e.g., translating nicht into not with $p(not|nicht)$.
- Distortion: modeled by $d(j|i, l_e, l_f)$, e.g., distortion of go to gehe with $d(4|2, 7, 6)$.

Distortion instead of alignment



Same translation, same alignment, but in a different way



Distortion instead of alignment

- The alignment function (Models 1 and 2) predicts foreign input word positions conditioned to English output word positions, i.e., from output to input.
- The distortion function (Model 3) predicts output word positions based on input word positions, i.e., from input to output.

Formulation of IBM Model 3

- Fertility: each input word f_i generates ϕ_i output words according to $n(\phi_i|f_i)$.
- NULL Token insertion: its number ϕ_0 depends on the number of output words generated by the input words.
  - Each generated word may insert a NULL token.
  - Number of generated words from foreign input words: $\sum_{i=1}^{l_f} \phi_i = l_e - \phi_0$
  - Probability of generating ϕ_0 words from the NULL token: $p(\phi_0) = \binom{l_e - \phi_0}{\phi_0} p_1^{\phi_0} p_0^{l_e - 2\phi_0}$

Formulation of IBM Model 3

Combining the four steps:

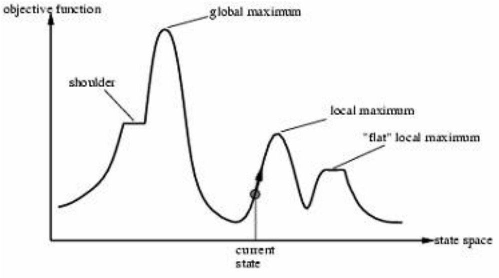
$$
\begin{aligned}
p(\mathbf{e}|\mathbf{f}) &= \sum_{a} p(e, a|f) \\
&= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \binom{l_e - \phi_0}{\phi_0} p_1^{\phi_0} p_0^{l_e - 2\phi_0} \\
&\times \prod_{j=1}^{l_f} \phi_i ! n(\phi_i | f_i) \\
&\times \prod_{j=1}^{l_e} t(e_j | f_{a(j)}) d(j | a(j), l_e, l_f)
\end{aligned}
$$

- This time we cannot reduce the complexity from exponential to polynomial.

Sampling the Alignment Space

- Training IBM Model 3 with the EM algorithm
  - The trick that reduces exponential complexity does not work anymore
  - → Not possible to exhaustively consider all alignments
- Two tasks:
  - Finding the most probable alignment by **hill climbing**
  - Sampling: collecting additional variations to calculate statistics

Hill climbing



http://www35.homepage.villanova.edu/abdo.achkar/csc8530/proj.htm

Hill climbing

- Finding the most probable alignment by hill climbing
  - start with initial alignment (e.g., Model 2)
  - change alignments for individual words
  - keep change if it has higher probability
  - continue until convergence

Sampling

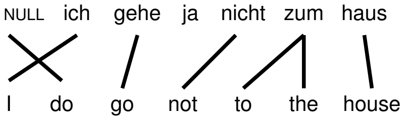
- Collecting variations to collect statistics
  - all alignments found during **hill climbing**
  - neighboring alignments that differ by a move or a swap

IBM Model 4

- Better reordering model
- Reordering in IBM Model 2 and 3
  - recall: $d(j|i, l_e, lf)$
  - for large sentences (large l_f and l_e), sparse and unreliable statistics
  - phrases tend to move together
- Relative reordering model: relative to previously translated words (cepts)

IBM Model 4: Cepts

Foreign words with non-zero fertility forms cepts (here 5 cepts)



cept π_i	π_1	π_2	π_3	π_4	π_5
foreign position $[i]$	1	2	4	5	6
foreign word $f_{[i]}$	ich	gehe	nicht	zum	haus
English words $\{e_j\}$	I	go	not	to,the	house
English positions $\{j\}$	1	4	3	5,6	7
center of cept $\odot_i$	1	4	3	6	7

The center of a cept is defined as the ceiling of the average of the output word positions for that cept.

IBM Model 4: Relative Distortion

j	1	2	3	4	5	6	7
e_j	I	do	not	go	to	the	house
in cept $\pi_{i,k}$	$\pi_{1,0}$	$\pi_{0,0}$	$\pi_{3,0}$	$\pi_{2,0}$	$\pi_{4,0}$	$\pi_{4,1}$	$\pi_{5,0}$
$\odot_{i-1}$	0	-	4	1	3	-	6
$j - \odot_{i-1}$	$+1$	-	-1	$+3$	$+2$	-	$+1$
distortion	$d_1(+1)$	1	$d_1(-1)$	$d_1(+3)$	$d_1(+2)$	$d_{>1}(+1)$	$d_1(+1)$

- Center $\odot_i$ of a cept π_i is ceiling(avg(j))
- Three cases:
  - NULL generated words: uniform distribution
  - first word of a cept: $d_1(j - \odot_{i-1})$
  - next words of a cept: $d_{>1}(j - \pi_{i,k-1})$

Training Model 4

- We also use the hill climbing strategy (just like in Model 3)
- But due to the complexity of the model (distortion, cepts), a hill-climbing method based on Model 3 probabilities is proposed.

IBM Model 5

- IBM Models 1–4 are *deficient*
  - some impossible translations have positive probability
  - multiple output words may be placed in the same position
  - → probability mass is wasted
- IBM Model 5 fixes deficiency by keeping track of **vacant word positions** (available positions)

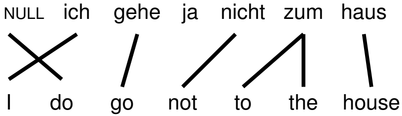
Formalization of IBM Model 5

- Number of vacancies in the English output interval $[1; j]$: v_j
- Distortion probabilities:

$$\text{for initial word in cept:} \quad d_1(v_j | \mathcal{B}(e_j), v_{\odot_{i-1}}, v_{max})$$
$$\text{for additional words:} \quad d_{>1}(v_j - v_{\pi_{i,k-1}} | \mathcal{B}(e_j), v_{max'})$$

- Maximum number of available vacancies: v_{max}
- Number of vacancies at the position of the previously placed English word: $v_{\pi_{i,k-1}}$

IBM Model 5: Example



cept		vacancies							parameters for d_1			
$f_{[i]}$	$\pi_{i,k}$	v_1 I	v_2 do	v_3 not	v_4 go	v_5 to	v_6 the	v_7 house	j	v_j	v_{max}	$v_{\odot_{i-1}}$
NULL	$\pi_{0,1}$	1	**2**	3	4	5	6	7	2	-	-	-
ich	$\pi_{1,1}$	**1**	-	2	3	4	5	6	1	1	6	0
gehe	$\pi_{2,1}$	-	-	1	**2**	3	4	5	4	2	5	0
nicht	$\pi_{3,1}$	-	-	**1**	-	2	3	4	3	1	4	1
zum	$\pi_{4,1}$	-	-	-	-	**1**	2	-	5	1	2	0
	$\pi_{4,2}$	-	-	-	-	-	**1**	2	6	-	-	-
haus	$\pi_{5,1}$	-	-	-	-	-	-	**1**	7	1	1	0

Conclusion

- IBM Models were the pioneering models in statistical machine translation
- Introduced important concepts
  - generative model
  - EM training
  - reordering models
- No longer state of the art models for machine translation...
- ... but still in common use for word alignment (e.g., GIZA++ toolkit)

Summary

- Expectation Maximization (EM) Algorithm
- Noisy Channel Model
- IBM Models 1–5
  - IBM Model 1: lexical translation
  - IBM Model 2: alignment model
  - IBM Model 3: fertility
  - IBM Model 4: relative alignment model
  - IBM Model 5: deficiency

Suggested reading

- Statistical Machine Translation, Philipp Koehn (section 4.1-4.4).