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 only uses lexical translation

Translation probability

for a foreign sentence \( f = (f_1, \ldots, f_{l_f}) \) of length \( l_f \)

to an English sentence \( e = (e_1, \ldots, e_{l_e}) \) of length \( l_e \)

with an alignment of each English word \( e_j \) to a foreign word \( f_i \)

according to the alignment function \( a : j \rightarrow i \)

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

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

- The right side is the product over the \textbf{lexical translation probabilities} for all $l_e$ generated output words $e_j$. 
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 \(\epsilon\) is a normalization constant.
Example

\[
p(e, a|f) = \frac{\epsilon}{5^4} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein})
\]
\[
= \frac{\epsilon}{5^4} \times 0.7 \times 0.8 \times 0.8 \times 0.4
\]
\[
= 0.0028\epsilon
\]
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*,
    → we could estimate the *parameters* of our generative model
  - if we had the *parameters*,
    → 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

... 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
Parameter estimation from the aligned corpus
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
We need to compute \( p(a|e, f) \)

Applying the chain rule:

\[
p(a|e, f) = \frac{p(e, a|f)}{p(e|f)}
\]

We already have the formula for \( p(e, a|f) \) (definition of Model 1)
We need to compute $p(e|f)$

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

$$= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} p(e, a|f)$$

$$= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \epsilon \frac{\epsilon}{(l_f + 1)^l_e} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$
IBM Model 1 and EM: Expectation Step

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

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

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

- 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

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

\[ = t(e_1|f_0) \cdot t(e_2|f_0) + t(e_1|f_0) \cdot t(e_2|f_1) + t(e_1|f_0) \cdot t(e_2|f_2) + \]
\[ + t(e_1|f_1) \cdot t(e_2|f_0) + t(e_1|f_1) \cdot t(e_2|f_1) + t(e_1|f_1) \cdot t(e_2|f_2) + \]
\[ + t(e_1|f_2) \cdot t(e_2|f_0) + t(e_1|f_2) \cdot t(e_2|f_1) + t(e_1|f_2) \cdot t(e_2|f_2) = \]

\[ = t(e_1|f_0) \cdot (t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) + \]
\[ + t(e_1|f_1) \cdot (t(e_2|f_1) + t(e_2|f_1) + t(e_2|f_2)) + \]
\[ + t(e_1|f_2) \cdot (t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2)) = \]

\[ = (t(e_1|f_0) + t(e_1|f_1) + t(e_1|f_2)) \cdot (t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2)) \]
IBM Model 1 and EM: Expectation Step

- Combine what we have:

\[
p(a|e, f) = \frac{p(e, a|f)}{p(e|f)} = \frac{\epsilon \prod_{j=1}^{l_e} t(e_j|f_{a(j)})}{\epsilon \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)} = \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)}
\]
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 \( f \):

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

With the same simplification as before:

\[
c(e|f; e, f) = \frac{t(e|f)}{\sum_{i=0}^{l_f} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e, e_j) \sum_{i=0}^{l_f} \delta(f, f_i)
\]
After collecting these counts over a corpus, we can estimate the model:

\[
t(e|f; e, f) = \frac{\sum (e, f) c(e|f; e, f))}{\sum_e \sum (e, f) c(e|f; e, f))}
\]
**IBM Model 1 and EM: Pseudo-code**

**Input:** set of sentence pairs \((e, f)\)

**Output:** translation prob. \(t(e|f)\)

1: initialize \(t(e|f)\) uniformly

2: while not converged do

3: // initialize

4: \(\text{count}(e|f) = 0\) for all \(e, f\)

5: \(\text{total}(f) = 0\) for all \(f\)

6: for all sentence pairs \((e, f)\) do

7: // compute normalization

8: for all words \(e\) in \(e\) do

9: \(s\text{-total}(e) = 0\)

10: for all words \(f\) in \(f\) do

11: \(s\text{-total}(e) += t(e|f)\)

12: end for

13: end for

14: // collect counts

15: for all words \(e\) in \(e\) do

16: for all words \(f\) in \(f\) do

17: \(\text{count}(e|f) += \frac{t(e|f)}{s\text{-total}(e)}\)

18: \(\text{total}(f) += \frac{t(e|f)}{s\text{-total}(e)}\)

19: end for

20: end for

21: end for

22: // estimate probabilities

23: for all foreign words \(f\) do

24: for all English words \(e\) do

25: \(t(e|f) = \frac{\text{count}(e|f)}{\text{total}(f)}\)

26: end for

27: end for

28: end while
Convergence

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>initial</th>
<th>1st it.</th>
<th>2nd it.</th>
<th>3rd it.</th>
<th>...</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>das</td>
<td>0.25</td>
<td>0.5</td>
<td>0.6364</td>
<td>0.7479</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>book</td>
<td>das</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1208</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>house</td>
<td>das</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1313</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>the</td>
<td>buch</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1208</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>book</td>
<td>buch</td>
<td>0.25</td>
<td>0.5</td>
<td>0.6364</td>
<td>0.7479</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>buch</td>
<td>0.25</td>
<td>0.25</td>
<td>0.1818</td>
<td>0.1313</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>book</td>
<td>ein</td>
<td>0.25</td>
<td>0.5</td>
<td>0.4286</td>
<td>0.3466</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>ein</td>
<td>0.25</td>
<td>0.5</td>
<td>0.5714</td>
<td>0.6534</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>haus</td>
<td>0.25</td>
<td>0.5</td>
<td>0.4286</td>
<td>0.3466</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>house</td>
<td>haus</td>
<td>0.25</td>
<td>0.5</td>
<td>0.5714</td>
<td>0.6534</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>
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:

\[ p(\text{the book}|\text{das Buch}) = \frac{\epsilon}{2^2} (0.25 + 0.25)(0.25 + 0.25) = 0.0625\epsilon \]
Convergence of EM

After the first iteration:

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

\[
p(\text{the book}|\text{das Buch}) = \frac{\epsilon}{2^2} (0.5 + 0.25)(0.25 + 0.5) = 0.140625\epsilon
\]
Convergence of EM

This will ultimately converge to:

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

\[ p(\text{the book}|\text{das Buch}) = \frac{\epsilon}{2^2}(1 + 0)(0 + 1) = 0.25\epsilon \]
Perplexity

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

\[ \log_2 PP = - \sum_{s} \log_2 p(e_s | f_s) \]
**Example ($\epsilon=1$)**

<table>
<thead>
<tr>
<th></th>
<th>initial</th>
<th>1st it.</th>
<th>2nd it.</th>
<th>3rd it.</th>
<th>...</th>
<th>final</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p(\text{the haus}</td>
<td>\text{das haus})$</td>
<td>0.0625</td>
<td>0.1875</td>
<td>0.1905</td>
<td>0.1913</td>
<td>...</td>
</tr>
<tr>
<td>$p(\text{the book}</td>
<td>\text{das buch})$</td>
<td>0.0625</td>
<td>0.1406</td>
<td>0.1790</td>
<td>0.2075</td>
<td>...</td>
</tr>
<tr>
<td>$p(\text{a book}</td>
<td>\text{ein buch})$</td>
<td>0.0625</td>
<td>0.1875</td>
<td>0.1907</td>
<td>0.1913</td>
<td>...</td>
</tr>
<tr>
<td><strong>perplexity</strong></td>
<td>4095</td>
<td>202.3</td>
<td>153.6</td>
<td>131.6</td>
<td>...</td>
<td>113.8</td>
</tr>
</tbody>
</table>

- 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.
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):

\[
p(e) = p(e_1, e_2, \ldots, e_n) \\
= p(e_1)p(e_2|e_1)\ldots p(e_n|e_1, e_2, \ldots, e_{n-1}) \\
\approx p(e_1)p(e_2|e_1)\ldots p(e_n|e_{n-2}, e_{n-1})
\]
Statistics can be computed based on both the English dataset of the parallel corpus.
But also on any text resource in this language (English).
We would like to integrate a language model. We look for the best translation $e$ for the input foreign sentence $f$. Use Bayes rule to include $p(e)$:

$$\text{argmax}_e \ p(e|f) = \text{argmax}_e \ \frac{p(f|e) \ p(e)}{p(f)}$$

$$= \text{argmax}_e \ p(f|e) \ p(e)$$
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 \( e \))
<table>
<thead>
<tr>
<th>IBM Model 1</th>
<th>lexical translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Model 2</td>
<td>adds absolute reordering model</td>
</tr>
<tr>
<td>IBM Model 3</td>
<td>adds fertility model</td>
</tr>
<tr>
<td>IBM Model 4</td>
<td>relative reordering model</td>
</tr>
<tr>
<td>IBM Model 5</td>
<td>fixes deficiency</td>
</tr>
</tbody>
</table>
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 \( a : j \rightarrow i \)

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

- Parameter \( \epsilon \) is a normalization constant
Adding a model of alignment:

IBM Model 2
Alignment probability

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

\[ a(i|j, l_e, l_f) \]
We have a two-step process:

- **Lexical translation step**: Translation probability (e.g., $t(is|ist)$)
- **Alignment step**: Alignment probability (e.g., $a(2|5, 6, 5)$)
IBM Model 2

- Putting everything together

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

- EM training of this model works the same way as IBM Model 1
IBM Model 2: Expectation Step

\[ p(e|f) = \sum_a p(e, a|f) \]

\[ = \epsilon \sum_{a(1)=0}^{l_f} \ldots \sum_{a(l_e)=0}^{l_f} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})a(a(j)|j, l_e, l_f) \]

\[ = \epsilon \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)a(i|j, l_e, l_f) \]

- We use the same trick, just like Model 1
IBM Model 2: Maximization Step

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

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

- and the counts for **alignments**:

$$c(i|j; l_e, l_f e, f) = \frac{t(e_j|f_i)a(i|j, l_e, l_f)}{\sum_{i'_{l_f}}^l f i' t(e|f i') a(i'|j, l_e, l_f)}$$
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. ich
2. gehe
3. ja
4. nicht
5. zum
6. haus

ich gehe nicht zum haus

ich gehe zum haus

ich NULL gehe nicht zum haus

I do not go to the house

I do go to the house

fertility step

NULL insertion step

lexical translation step

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

$$n(1|haus) \sim 1$$
$$n(2|zum) \sim 1$$
$$n(0|ja) \sim 1$$
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
IBM Model 3 - four-step process

1. **Fertility step**

2. **NULL insertion step**

3. **Lexical translation step**

4. **Distortion step**

```
ich  gehe  ja  nicht  zum  haus
  ↓    ↓     ↓    ↓        ↓
ich  gehe  nicht  zum  zum  haus
      ↓    ↓     ↓    ↓        ↓
ich  NULL  gehe  nicht  zum  zum  haus
            ↓    ↓     ↓    ↓        ↓
I do  not  go  to  the  house
```

```
I do  go  not  to  the  house
```
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.
Fertility: each input word $f_i$ generates $\phi_i$ output words according to $n(\phi_i|f_i)$.

NULL Token insertion: its number $\phi_0$ 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 $\phi_0$ 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:

\[
p(e|f) = \sum_{a} p(e, a|f)
\]

\[
= \sum_{a(1)=0}^{l_f} \ldots \sum_{a(l_e)=0}^{l_f} \left( l_e - \phi_0 \right)^{p_1 \phi_0} p_0^{l_e-2\phi_0}
\]

\[
\times \prod_{j=1}^{l_f} \phi_i^n(\phi_i|f_i)
\]

\[
\times \prod_{j=1}^{l_e} t(e_j|f_a(j))d(j|a(j), l_e, l_f)
\]

- 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
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: \( d(j|i, l_e, l_f) \)
  - 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)

\[
\begin{array}{cccccc}
\text{NULL} & \text{ich} & \text{gehe} & \text{ja} & \text{nicht} & \text{zum} & \text{haus} \\
\text{I} & \text{do} & \text{go} & \text{not} & \text{to} & \text{the} & \text{house} \\
\end{array}
\]

<table>
<thead>
<tr>
<th>cept ( \pi_i )</th>
<th>( \pi_1 )</th>
<th>( \pi_2 )</th>
<th>( \pi_3 )</th>
<th>( \pi_4 )</th>
<th>( \pi_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>foreign position ([i])</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>foreign word (f_{[i]})</td>
<td>ich</td>
<td>gehe</td>
<td>nicht</td>
<td>zum</td>
<td>haus</td>
</tr>
<tr>
<td>English words ({e_j})</td>
<td>I</td>
<td>go</td>
<td>not</td>
<td>to, the</td>
<td>house</td>
</tr>
<tr>
<td>English positions ({j})</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>5,6</td>
<td>7</td>
</tr>
<tr>
<td>center of cept (\odot_i)</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$j$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_j$</td>
<td>I</td>
<td>do</td>
<td>not</td>
<td>go</td>
<td>to</td>
<td>the</td>
<td>house</td>
</tr>
<tr>
<td>in cept $\pi_{i,k}$</td>
<td>$\pi_{1,0}$</td>
<td>$\pi_{0,0}$</td>
<td>$\pi_{3,0}$</td>
<td>$\pi_{2,0}$</td>
<td>$\pi_{4,0}$</td>
<td>$\pi_{4,1}$</td>
<td>$\pi_{5,0}$</td>
</tr>
<tr>
<td>$\odot_{i-1}$</td>
<td>0</td>
<td>-</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>$j - \odot_{i-1}$</td>
<td>+1</td>
<td>-</td>
<td>-1</td>
<td>+3</td>
<td>+2</td>
<td>-</td>
<td>+1</td>
</tr>
<tr>
<td>distortion</td>
<td>$d_1(+1)$</td>
<td>1</td>
<td>$d_1(-1)$</td>
<td>$d_1(+3)$</td>
<td>$d_1(+2)$</td>
<td>$d_{&gt;1}(+1)$</td>
<td>$d_1(+1)$</td>
</tr>
</tbody>
</table>

- Center $\odot_i$ of a cept $\pi_i$ is $\text{ceiling}(\text{avg}(j))$
- 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 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:
  - for initial word in cept: $d_1(v_j | B(e_j), v_{\ominus_i-1}, v_{max})$
  - for additional words: $d_{>1}(v_j - v_{\pi_i,k-1} | B(e_j), v_{max'})$
- 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

- **German**: NULL ich gehe ja nicht zum haus
- **English**: I do not go to the house

---

<table>
<thead>
<tr>
<th>except</th>
<th>vacancies</th>
<th>parameters for $d_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f[i]$</td>
<td>$\pi_{i,k}$</td>
<td>$v_1$</td>
</tr>
<tr>
<td>NULL</td>
<td>$\pi_{0,1}$</td>
<td>1</td>
</tr>
<tr>
<td>ich</td>
<td>$\pi_{1,1}$</td>
<td>1</td>
</tr>
<tr>
<td>gehe</td>
<td>$\pi_{2,1}$</td>
<td>-</td>
</tr>
<tr>
<td>nicht</td>
<td>$\pi_{3,1}$</td>
<td>-</td>
</tr>
<tr>
<td>zum</td>
<td>$\pi_{4,1}$</td>
<td>-</td>
</tr>
<tr>
<td>$\pi_{4,2}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>haus</td>
<td>$\pi_{5,1}$</td>
<td>-</td>
</tr>
</tbody>
</table>
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).