Chapter 4

Word-based models

Statistical Machine Translation
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: In all lectures, we translate from a foreign language into English
Collect Statistics

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

<table>
<thead>
<tr>
<th>Translation of <em>Haus</em></th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>house</td>
<td>8,000</td>
</tr>
<tr>
<td>building</td>
<td>1,600</td>
</tr>
<tr>
<td>home</td>
<td>200</td>
</tr>
<tr>
<td>household</td>
<td>150</td>
</tr>
<tr>
<td>shell</td>
<td>50</td>
</tr>
</tbody>
</table>
Estimate Translation Probabilities

Maximum likelihood estimation

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

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

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\text{das} & \text{Haus} & \text{ist} & \text{klein} \\
\text{the} & \text{house} & \text{is} & \text{small} \\
\end{array}
\]

- Word positions are numbered 1–4
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 $a : i \rightarrow j$

- Example

  \[ a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4\} \]
Reordering

Words may be reordered during translation

\[ a : \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\} \]
One-to-Many Translation

A source word may translate into multiple target words

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\text{das} & \text{Haus} & \text{ist} & \text{klitzeklein} \\
\text{the} & \text{house} & \text{is} & \text{very} \quad \text{small} \\
1 & 2 & 3 & 4 & 5 \\
\end{array}
\]

\[a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4, 5 \rightarrow 4\}\]
Dropping Words

Words may be dropped when translated
(German article das is dropped)

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\text{das} & \text{Haus} & \text{ist} & \text{klein} \\
\text{house} & \text{is} & \text{small} \\
1 & 2 & 3 \\
\end{array}
\]

\[
a : \{1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4\}
\]
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 \to 1, 2 \to 2, 3 \to 3, 4 \to 0, 5 \to 4}
```
IBM Model 1

- Generative model: break up translation process into smaller steps
  - 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 \( a : j \rightarrow i \)

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

- parameter \( \epsilon \) is a normalization constant
### Example

| das   | t(e|f)   | Haus     | t(e|f) | ist     | t(e|f) | klein    | t(e|f) |
|-------|---------|----------|-------|---------|-------|----------|-------|
| the   | 0.7     | house    | 0.8   | is      | 0.8   | small    | 0.4   |
| that  | 0.15    | building | 0.16  | 's      | 0.16  | little   | 0.4   |
| which | 0.075   | home     | 0.02  | exists  | 0.02  | short    | 0.1   |
| who   | 0.05    | household| 0.015 | has     | 0.015 | minor    | 0.06  |
| this  | 0.025   | shell    | 0.005 | are     | 0.005 | petty    | 0.04  |

\[
p(e, a|f) = \frac{\epsilon}{4^3} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein})
\]

\[
= \frac{\epsilon}{4^3} \times 0.7 \times 0.8 \times 0.8 \times 0.4
\]

\[
= 0.0028\epsilon
\]
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, we 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

... 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 (pigeon hole principle)
EM Algorithm

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

/ / / / / X / / / / /

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

• Convergence

• Inherent hidden structure revealed by EM
EM Algorithm

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

/ / / X / / /

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

\[ p(\text{la} \mid \text{the}) = 0.453 \]
\[ p(\text{le} \mid \text{the}) = 0.334 \]
\[ p(\text{maison} \mid \text{house}) = 0.876 \]
\[ p(\text{bleu} \mid \text{blue}) = 0.563 \]

• Parameter estimation from the aligned corpus

Chapter 4: Word-Based Models 18
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

- 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
  
  \[ \text{la} \rightarrow \text{the} \quad \text{la} \rightarrow \text{the} \quad \text{la} \rightarrow \text{the} \quad \text{la} \rightarrow \text{the} \]
  
  \[ \text{maison} \rightarrow \text{house} \quad \text{maison} \rightarrow \text{house} \quad \text{maison} \rightarrow \text{house} \quad \text{maison} \rightarrow \text{house} \]

  \[ p(\text{e}, \text{a}|\text{f}) = 0.56 \quad p(\text{e}, \text{a}|\text{f}) = 0.035 \quad p(\text{e}, \text{a}|\text{f}) = 0.08 \quad p(\text{e}, \text{a}|\text{f}) = 0.005 \]

  \[ p(\text{a}|\text{e}, \text{f}) = 0.824 \quad p(\text{a}|\text{e}, \text{f}) = 0.052 \quad p(\text{a}|\text{e}, \text{f}) = 0.118 \quad p(\text{a}|\text{e}, \text{f}) = 0.007 \]

- Counts
  
  \[ c(\text{the}|\text{la}) = 0.824 + 0.052 \quad c(\text{house}|\text{la}) = 0.052 + 0.007 \]
  
  \[ c(\text{the}|\text{maison}) = 0.118 + 0.007 \quad c(\text{house}|\text{maison}) = 0.824 + 0.118 \]
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)
IBM Model 1 and EM: Expectation Step

- We need to compute $p(e|f)$

$$p(e|f) = \sum_a p(e, a|f)$$

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

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

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

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

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

- 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:

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

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

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

\[
= \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)}
\]
IBM Model 1 and EM: Maximization Step

- Now we have to collect counts

- 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:

$$c(e|f; e, f) = \frac{t(e|f)}{\sum_{i=0}^{l_f} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e, e_j) \sum_{i=0}^{l_f} \delta(f, f_i)$$
IBM Model 1 and EM: Maximization Step

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: Pseudocode**

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

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

1. initialize \(t(e \mid f)\) uniformly
2. while not converged do
3.   \(\text{// initialize}\)
4.   \(\text{count}(e \mid f) = 0 \text{ for all } e, f\)
5.   \(\text{total}(f) = 0 \text{ for all } f\)
6.   for all sentence pairs \((e, f)\) do
7.     \(\text{// compute normalization}\)
8.     for all words \(e\) in \(e\) do
9.       \(\text{s-total}(e) = 0\)
10.    for all words \(f\) in \(f\) do
11.       \(\text{s-total}(e) += t(e \mid f)\)
12.      end for
13.   end for
14. \(\text{// collect counts}\)
15.   for all words \(e\) in \(e\) do
16.     for all words \(f\) in \(f\) do
17.       \(\text{count}(e \mid f) += \frac{t(e \mid f)}{\text{s-total}(e)}\)
18.       \(\text{total}(f) += \frac{t(e \mid f)}{\text{s-total}(e)}\)
19.     end for
20.   end for
21.   end for
22. \(\text{// estimate probabilities}\)
23.   for all foreign words \(f\) do
24.     for all English words \(e\) do
25.       \(t(e \mid f) = \frac{\text{count}(e \mid f)}{\text{total}(f)}\)
26.     end for
27.   end for
28. end while
## Convergence

<table>
<thead>
<tr>
<th>$e$</th>
<th>$f$</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>
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>perplexity</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>
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

\[
p(e) = p(e_1, e_2, ..., e_n) \\
= p(e_1)p(e_2|e_1)...p(e_n|e_1, e_2, ..., e_{n-1}) \\
\simeq p(e_1)p(e_2|e_1)...p(e_n|e_{n-2}, e_{n-1})
\]
Noisy Channel Model

- We would like to integrate a language model
- Bayes rule

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

<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>

- 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
Reminder: IBM Model 1

- Generative model: break up translation process into smaller steps
  - 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 \( a : j \rightarrow i \)

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

- parameter \( \epsilon \) is a normalization constant
IBM Model 2

Adding a model of alignment

natürlich ist das haus klein
of course is the house small

das

of course the house is small

lexical translation step

alignment step
IBM Model 2

• Modeling alignment with an alignment probability distribution

• Translating foreign word at position \( i \) to English word at position \( j \):

\[
a(i|j, l_e, l_f)
\]

• 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
Interlude: HMM Model

- Words do not move independently of each other
  - they often move in groups
  → condition word movements on previous word

- HMM alignment model:

\[ p(a(j)|a(j-1), l_e) \]

- EM algorithm application harder, requires dynamic programming

- IBM Model 4 is similar, also conditions on word classes
IBM Model 3

Adding a model of fertility

1. ich
2. gehe
3. ja
4. nicht
5. zum
6. haus

fertility step

ich
gehe
nicht
zum
haus
Null insertion step

ich
NULL
gehe
nicht
zum
haus

lexical translation step

ich
NULL
gehe
nicht
zum
haus

l
do
not
go
to
the	house
distortion step

l
do
not
go
to
the	house
IBM Model 3: Fertility

- Fertility: number of English words generated by a foreign word
- Modelled by distribution $n(\phi|f)$
- Example:
  
  $n(1|\text{haus}) \approx 1$
  
  $n(2|\text{zum}) \approx 1$
  
  $n(0|\text{ja}) \approx 1$
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

- Finding the most probable alignment by hillclimbing
  - start with initial alignment
  - 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 hillclimbing
  - 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)

<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>
## 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>$\circ_{i-1}$</td>
<td>0</td>
<td>-</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>$j - \circ_{i-1}$</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 $\circ_i$ of a cept $\pi_i$ is $\text{ceiling(avg}(j))$

- Three cases:
  - Uniform for NULL generated words
  - First word of a cept: $d_1$
  - Next words of a cept: $d_{>1}$
Word Classes

• Some words may trigger reordering → condition reordering on words

  for initial word in cept: \(d_1(j - \odot_{[i-1]}|f_{[i-1]}, e_j)\)
  for additional words: \(d_{>1}(j - \Pi_{i,k-1}|e_j)\)

• Sparse data concerns → cluster words into classes

  for initial word in cept: \(d_1(j - \odot_{[i-1]}|A(f_{[i-1]}), B(e_j))\)
  for additional words: \(d_{>1}(j - \Pi_{i,k-1}|B(e_j))\)
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 vacancies (available positions)
Conclusion

• IBM Models were the pioneering models in statistical machine translation

• Introduced important concepts
  – generative model
  – EM training
  – reordering models

• Only used for niche applications as translation model

• ... but still in common use for word alignment (e.g., GIZA++ toolkit)
Word Alignment

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

Chapter 4: Word-Based Models
Word Alignment?

Is the English word does aligned to the German wohnt (verb) or nicht (negation) or neither?
Word Alignment?

<table>
<thead>
<tr>
<th></th>
<th>john</th>
<th>biss</th>
<th>ins</th>
<th>grass</th>
</tr>
</thead>
<tbody>
<tr>
<td>john</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kicked</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bucket</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How do the idioms *kicked the bucket* and *biss ins grass* match up? Outside this exceptional context, *bucket* is never a good translation for *grass*.
Measuring Word Alignment Quality

- Manually align corpus with *sure* ($S$) and *possible* ($P$) alignment points ($S \subseteq P$)

- Common metric for evaluation word alignments: Alignment Error Rate (AER)

$$AER(S, P; A) = \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

- AER = 0: alignment $A$ matches all sure, any possible alignment points

- However: different applications require different precision/recall trade-offs
Word Alignment with IBM Models

- 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 can not return multiple values for one input
    (no many-to-one mapping)

- Real word alignments have **many-to-many** mappings
Symmetrizing Word Alignments

- Intersection of GIZA++ bidirectional alignments
- Grow additional alignment points [Och and Ney, CompLing2003]
Growing heuristic

grow-diag-final(e2f,f2e)
1: neighboring = {(-1,0),(0,-1),(1,0),(0,1),(-1,-1),(-1,1),(1,-1),(1,1)}
2: alignment A = intersect(e2f,f2e); grow-diag(); final(e2f); final(f2e);

grow-diag()
1: while new points added do
2: for all English word e ∈ [1...en], foreign word f ∈ [1...fn], (e, f) ∈ A do
3: for all neighboring alignment points (e_new, f_new) do
4: if (e_new unaligned OR f_new unaligned) AND (e_new, f_new) ∈ union(e2f,f2e) then
5: add (e_new, f_new) to A
6: end if
7: end for
8: end for
9: end while

final()
1: for all English word e_new ∈ [1...en], foreign word f_new ∈ [1...fn] do
2: if (e_new unaligned OR f_new unaligned) AND (e_new, f_new) ∈ union(e2f,f2e) then
3: add (e_new, f_new) to A
4: end if
5: end for
More Recent Work on Symmetrization

- Symmetrize after each iteration of IBM Models [Matusov et al., 2004]
  - run one iteration of E-step for each direction
  - symmetrize the two directions
  - count collection (M-step)

- Use of posterior probabilities in symmetrization
  - generate n-best alignments for each direction
  - calculate how often an alignment point occurs in these alignments
  - use this posterior probability during symmetrization
Link Deletion / Addition Models

• Link deletion [Fossum et al., 2008]
  – start with union of IBM Model alignment points
  – delete one alignment point at a time
  – uses a neural network classifiers that also considers aspects such as how useful the alignment is for learning translation rules

• Link addition [Ren et al., 2007] [Ma et al., 2008]
  – possibly start with a skeleton of highly likely alignment points
  – add one alignment point at a time
Discriminative Training Methods

- Given some annotated training data, supervised learning methods are possible

- Structured prediction
  - not just a classification problem
  - solution structure has to be constructed in steps

- Many approaches: maximum entropy, neural networks, support vector machines, conditional random fields, MIRA, ...

- Small labeled corpus may be used for parameter tuning of unsupervised aligner
  [Fraser and Marcu, 2007]
Better Generative Models

• Aligning phrases
  – joint model [Marcu and Wong, 2002]
  – problem: EM algorithm likes really long phrases

• Fraser’s LEAF
  – decomposes word alignment into many steps
  – similar in spirit to IBM Models
  – includes step for grouping into phrase
Summary

- Lexical translation
- Alignment
- 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
- Word Alignment