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)

<table>
<thead>
<tr>
<th>Translation of Haus</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

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

• 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

\[
\begin{align*}
&\text{das Haus ist klein} \\
&\text{the house is small}
\end{align*}
\]

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

One-to-many translation

- A source word may translate into multiple target words

\[
\begin{align*}
&\text{das Haus ist klitzeklein} \\
&\text{the house is very small}
\end{align*}
\]

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

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

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

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

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

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 & 4 \\
\text{NULL} & \text{das} & \text{Haus} & \text{ist} & \text{klein} \\
\text{the} & \text{house} & \text{is} & \text{just} & \text{small} \\
\end{array}
\]

\[a : \{1 \to 1, 2 \to 2, 3 \to 3, 4 \to 0, 5 \to 4\}\]
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 \( f = (f_1, ..., f_{l_f}) \) 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 \( f_i \) 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**

<table>
<thead>
<tr>
<th>das</th>
<th>Haus</th>
<th>ist</th>
<th>klein</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>t(e</td>
<td>f)</td>
<td>e</td>
</tr>
<tr>
<td>the</td>
<td>0.7</td>
<td>is</td>
<td>0.8</td>
</tr>
<tr>
<td>that</td>
<td>0.15</td>
<td>’s</td>
<td>0.16</td>
</tr>
<tr>
<td>which</td>
<td>0.075</td>
<td>exists</td>
<td>0.02</td>
</tr>
<tr>
<td>who</td>
<td>0.05</td>
<td>has</td>
<td>0.015</td>
</tr>
<tr>
<td>this</td>
<td>0.025</td>
<td>are</td>
<td>0.005</td>
</tr>
</tbody>
</table>

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

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

\[
= 0.0028 \epsilon
\]
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*, would could estimate *model*
  - if we had *model*, we could fill in the *gaps in the data*

- **Expectation Maximization (EM)** in a nutshell
  - initialize model parameters (e.g. uniform)
  - assign probabilities to the missing data
  - 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

• After another iteration

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

EM algorithm

• Convergence

• Inherent hidden structure revealed by EM
EM algorithm

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

[Diagram: Alignment between French and English sentences]

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

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

- 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

Probability

\[
p(\text{the}|\text{la}) = 0.7 \quad p(\text{house}|\text{la}) = 0.05 \\
p(\text{the}|\text{maison}) = 0.1 \quad p(\text{house}|\text{maison}) = 0.8
\]

Alignments

\[
p(\text{e}, a|\text{f}) = 0.56 \quad p(\text{e}, a|\text{f}) = 0.035 \\
p(\text{e}, a|\text{f}) = 0.08 \quad p(\text{e}, a|\text{f}) = 0.005 \\
p(\text{e}, a|\text{f}) = 0.824 \quad p(\text{e}, a|\text{f}) = 0.052 \\
p(\text{e}, a|\text{f}) = 0.118 \quad p(\text{e}, a|\text{f}) = 0.007
\]

Counts

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

- 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)

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

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

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

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

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

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

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

Philipp Koehn EMNLP Lecture 15 25 February 2008

---

The trick

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

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

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

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

Philipp Koehn EMNLP Lecture 15 25 February 2008

---

The trick (case \(l_e = l_f = 2\))

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

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

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

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

Philipp Koehn EMNLP Lecture 15 25 February 2008
IBM Model 1 and EM: Expectation Step

• Combine what we have:

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

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

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

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:

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

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

<table>
<thead>
<tr>
<th>IBM Model 1</th>
<th>lexical translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Model 2</td>
<td>adds absolute <strong>reordering model</strong></td>
</tr>
<tr>
<td>IBM Model 3</td>
<td>adds <strong>fertility model</strong></td>
</tr>
<tr>
<td>IBM Model 4</td>
<td>relative reordering model</td>
</tr>
<tr>
<td>IBM Model 5</td>
<td>fixes <strong>deficiency</strong></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

IBM Model 4

Mary did not slap the green witch

Mary not slap slap slap slap the green witch

Mary not slap slap slap slap NULL the green witch

Maria no daba una bofetada a la verde bruja

Maria no daba una bofetada a la bruja verde

\[ n(3|\text{slap}) \]
\[ p-\text{null} \]
\[ t(la|\text{the}) \]
\[ d(4|4) \]