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 <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

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

• 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

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

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

One-to-many translation

- A source word may translate into multiple target words

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

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

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

```
das    Haus    ist    klein
/      /       /       /
house  is      small
```

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

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

```
0    1    2    3    4
NULL  das  Haus  ist  klein
```

```
the    house    is    just    small
\[ a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 0, 5 \rightarrow 4\} \]```
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

das Haus ist klein

t(e|f)

<table>
<thead>
<tr>
<th>das</th>
<th>Haus</th>
<th>ist</th>
<th>klein</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>house</td>
<td>is</td>
<td>small</td>
</tr>
<tr>
<td>0.7</td>
<td>0.8</td>
<td>0.8</td>
<td>0.4</td>
</tr>
<tr>
<td>that</td>
<td>building</td>
<td>'s</td>
<td>little</td>
</tr>
<tr>
<td>0.15</td>
<td>0.16</td>
<td>0.16</td>
<td>0.4</td>
</tr>
<tr>
<td>which</td>
<td>home</td>
<td>exists</td>
<td>short</td>
</tr>
<tr>
<td>0.075</td>
<td>0.02</td>
<td>0.02</td>
<td>0.1</td>
</tr>
<tr>
<td>who</td>
<td>household</td>
<td>has</td>
<td>minor</td>
</tr>
<tr>
<td>0.05</td>
<td>0.015</td>
<td>0.015</td>
<td>0.06</td>
</tr>
<tr>
<td>this</td>
<td>shell</td>
<td>are</td>
<td>petty</td>
</tr>
<tr>
<td>0.025</td>
<td>0.005</td>
<td>0.005</td>
<td>0.04</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*, we 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

---

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

... 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

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

- **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**

  \[
  \begin{align*}
  \text{la} & \rightarrow \text{the} & \text{la} & \rightarrow \text{the} & \text{la} & \rightarrow \text{the} & \text{la} & \rightarrow \text{the} \\
  \text{maison} & \rightarrow \text{house} & \text{maison} & \rightarrow \text{house} & \text{maison} & \rightarrow \text{house} & \text{maison} & \rightarrow \text{house}
  \end{align*}
  \]

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

- **Counts**

  \[
  \begin{align*}
  c(\text{the}|\text{la}) &= 0.824 + 0.052 \\
  c(\text{the}|\text{maison}) &= 0.118 + 0.007
  \end{align*}
  \]

  \[
  \begin{align*}
  c(\text{house}|\text{la}) &= 0.052 + 0.007 \\
  c(\text{house}|\text{maison}) &= 0.824 + 0.118
  \end{align*}
  \]

---

IBM Model 1 and EM: Expectation Step

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

  \[
  p(\text{a}|\text{e}, \text{f}) = \frac{p(\text{e}, \text{a}|\text{f})}{p(\text{e}|\text{f})}
  \]
- We already have the formula for \( p(\text{e}, \text{a}|\text{f}) \) (definition of Model 1)
IBM Model 1 and EM: Expectation Step

• We need to compute $p(e|f)$

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

$$= \sum_a^{l_f} \cdots \sum_a^{l_{l_e}} p(e, a|f)$$

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

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

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

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

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

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

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

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

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

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

---

IBM Model 1 and EM: Expectation Step

- Combine what we have:

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

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

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

- After collecting these counts over a corpus, we can estimate the model:
  \[
  t(e|f; e, f) = \frac{\sum_{(e,f)} c(e|f; e, f)}{\sum_f \sum_{(e,f)} c(e|f; e, f)}
  \]
IBM Model 1 and EM: Pseudocode

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

\[
\text{do until convergence} \\
\text{set count}(e|f) \text{ to 0 for all } e,f \\
\text{set total}(f) \text{ to 0 for all } f \\
\text{for all sentence pairs } (e_s,f_s) \\
\text{for all words } e \text{ in } e_s \\
\quad \text{total}_s(e) = 0 \\
\text{for all words } f \text{ in } f_s \\
\quad \text{total}_s(e) += t(e|f) \\
\text{for all words } e \text{ in } e_s \\
\quad \text{for all words } f \text{ in } f_s \\
\quad \text{count}(e|f) += t(e|f) / \text{total}_s(e) \\
\quad \text{total}(f) += t(e|f) / \text{total}_s(e) \\
\text{for all } f \\
\text{for all } e \\
\quad t(e|f) = \text{count}(e|f) / \text{total}(f)
\]

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 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
## Word alignment

- Notion of **word alignment** valuable
- Shared task at NAACL 2003 and ACL 2005 workshops
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)

- But we need *many-to-many* mappings

Symmetrizing word alignments

- Intersection of GIZA++ bidirectional alignments
Symmetrizing word alignments

Maria no daba una bofetada a la bruja verde

• Grow additional alignment points [Och and Ney, CompLing2003]

Growing heuristic

GROW-DIAG-FINAL(e2f,f2e):
neighboring = ( (-1,0), (0,-1), (1,0), (0,1), (-1,-1), (-1,1), (1,-1), (1,1) )
alignment = intersect(e2f,f2e);
GROW-DIAG(); FINAL(e2f); FINAL(f2e);

GROW-DIAG():
iterate until no new points added
for english word e = 0 ... en
  for foreign word f = 0 ... fn
    if ( e aligned with f )
      for each neighboring point ( e-new, f-new ):
        if ( e-new not aligned and f-new not aligned ) and
            ( e-new, f-new ) in union( e2f, f2e )
          add alignment point ( e-new, f-new )

FINAL(a):
for english word e-new = 0 ... en
  for foreign word f-new = 0 ... fn
  if ( e-new not aligned or f-new not aligned ) and
      ( e-new, f-new ) in alignment a
    add alignment point ( e-new, f-new )
More Recent Work

• Symmetrization during training
  – symmetrize after each iteration of IBM Models
  – integrate symmetrization into models

• Discriminative training methods
  – supervised learning based on labeled data
  – semi-supervised learning with limited labeled data

• Better generative models
  – see talk by Alexander Fraser