IBM Model 1 and the EM Algorithm

Philipp Koehn

10 September 2020
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
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

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

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

A source word may translate into multiple target words

\[
das \text{ Haus ist klitzeklein} \rightarrow \text{the house is very small}
\]

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

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

\[
\begin{align*}
1 & \quad 2 & \quad 3 & \quad 4 \\
\text{das} & \quad \text{Haus} & \quad \text{ist} & \quad \text{klein} \\
/ & \quad / & \quad / & \\
\text{house} & \quad \text{is} & \quad \text{small} \\
1 & \quad 2 & \quad 3
\end{align*}
\]

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

\[
a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 0, 5 \rightarrow 4\}
\]
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

<table>
<thead>
<tr>
<th>das</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>e</em></td>
<td>*t(e</td>
</tr>
<tr>
<td>the</td>
<td>0.7</td>
</tr>
<tr>
<td>that</td>
<td>0.15</td>
</tr>
<tr>
<td>which</td>
<td>0.075</td>
</tr>
<tr>
<td>who</td>
<td>0.05</td>
</tr>
<tr>
<td>this</td>
<td>0.025</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Haus</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>e</em></td>
<td>*t(e</td>
</tr>
<tr>
<td>house</td>
<td>0.8</td>
</tr>
<tr>
<td>building</td>
<td>0.16</td>
</tr>
<tr>
<td>home</td>
<td>0.02</td>
</tr>
<tr>
<td>household</td>
<td>0.015</td>
</tr>
<tr>
<td>shell</td>
<td>0.005</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ist</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>e</em></td>
<td>*t(e</td>
</tr>
<tr>
<td>is</td>
<td>0.8</td>
</tr>
<tr>
<td>’s</td>
<td>0.16</td>
</tr>
<tr>
<td>exists</td>
<td>0.02</td>
</tr>
<tr>
<td>has</td>
<td>0.015</td>
</tr>
<tr>
<td>are</td>
<td>0.005</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>klein</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>e</em></td>
<td>*t(e</td>
</tr>
<tr>
<td>small</td>
<td>0.4</td>
</tr>
<tr>
<td>little</td>
<td>0.4</td>
</tr>
<tr>
<td>short</td>
<td>0.1</td>
</tr>
<tr>
<td>minor</td>
<td>0.06</td>
</tr>
<tr>
<td>petty</td>
<td>0.04</td>
</tr>
</tbody>
</table>

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

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

\[
= 0.0028\epsilon
\]
finding translations
Centauri-Arcturan Parallel Text

1a. ok-voon ororok sprok .
1b. at-voon bichat dat .

2a. ok-drubel ok-voon anok plok sprok .
2b. at-drubel at-voon pippat rrat dat .

3a. erok sprok izok hihok ghirok .
3b. totat dat arrat vat hilat .

4a. ok-voon anok drok brok jok .
4b. at-voon krat pippat sat lat .

5a. wiwok farok izok stok .
5b. totat jjat quat cat .

6a. lalok sprok izok jok stok .
6b. wat dat kratquat cat .

7a. lalok farok ororok lalok sprok izok enemok .
7b. wat jjat bichat wat dat vat eneat .

8a. lalok brok anok plok nok .
8b. iat lat pippat rrat nnat .

9a. wiwok nok izok kantok ok-yurp .
9b. totat nnat quat oloat at-yurp .

10a. lalok mok nok yorok ghirok clok .
10b. wat nnat gat mat bat hilat .

11a. lalok nok crrrok hihok yorok zanzanok .
11b. wat nnat arrat mat zanzanat .

12a. lalok rarok nok izok hihok mok .
12b. wat nnat forat arrat vat gat .

Translation challenge: farok crrrok hihok yorok clok kantok ok-yurp

(from Knight (1997): Automating Knowledge Acquisition for Machine Translation)
em algorithm
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

... 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 ...

... 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
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**

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

- **Counts**
  \[ c(\text{the} | \text{la}) = 0.824 + 0.052 \]
  \[ c(\text{the} | \text{maison}) = 0.118 + 0.007 \]
  \[ c(\text{house} | \text{la}) = 0.052 + 0.007 \]
  \[ c(\text{house} | \text{maison}) = 0.824 + 0.118 \]
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)
- We need to compute \( p(e|f) \)

\[
p(e|f) = \sum_a p(e, a|f) \\
= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} p(e, a|f) \\
= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f} \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})
\]
IBM Model 1 and EM: Expectation Step

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

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

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

- Note the trick in the last line
  - removes the need for an exponential number of products
  \[ \rightarrow \] this makes IBM Model 1 estimation tractable
The Trick

(case $l_e = l_f = 2$)

\[
\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} = \frac{\epsilon}{3^2} \prod_{j=1}^{2} t(e_j | f_{a(j)}) = \\
= t(e_1 | f_0) t(e_2 | f_0) + t(e_1 | f_0) t(e_2 | f_1) + t(e_1 | f_0) t(e_2 | f_2) + \\
+ t(e_1 | f_1) t(e_2 | f_0) + t(e_1 | f_1) t(e_2 | f_1) + t(e_1 | f_1) t(e_2 | f_2) + \\
+ t(e_1 | f_2) t(e_2 | f_0) + t(e_1 | f_2) t(e_2 | f_1) + t(e_1 | f_2) t(e_2 | f_2) = \\
= t(e_1 | f_0) (t(e_2 | f_0) + t(e_2 | f_1) + t(e_2 | f_2)) + \\
+ t(e_1 | f_1) (t(e_2 | f_1) + t(e_2 | f_1) + t(e_2 | f_2)) + \\
+ t(e_1 | f_2) (t(e_2 | f_2) + t(e_2 | f_1) + t(e_2 | f_2)) = \\
= (t(e_1 | f_0) + t(e_1 | f_1) + t(e_1 | f_2)) (t(e_2 | f_2) + t(e_2 | f_1) + t(e_2 | f_2))
\]
• Combine what we have:

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

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

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

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

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

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

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

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

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

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

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

Philipp Koehn

Machine Translation: IBM Model 1 and the EM Algorithm

10 September 2020
## 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>
Higher IBM Models

<table>
<thead>
<tr>
<th>IBM Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>lexical translation</td>
</tr>
<tr>
<td>Model 2</td>
<td>adds absolute reordering model</td>
</tr>
<tr>
<td>Model 3</td>
<td>adds fertility model</td>
</tr>
<tr>
<td>Model 4</td>
<td>relative reordering model</td>
</tr>
<tr>
<td>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
Word Alignment

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

<table>
<thead>
<tr>
<th>michael</th>
<th>geht</th>
<th>davon</th>
<th>aus</th>
<th>dass</th>
<th>er</th>
<th>im</th>
<th>haus</th>
<th>bleibt</th>
</tr>
</thead>
<tbody>
<tr>
<td>assumes</td>
<td>that</td>
<td>he</td>
<td>will</td>
<td>stay</td>
<td>in</td>
<td>the</td>
<td>house</td>
<td></td>
</tr>
</tbody>
</table>
Word Alignment?

<table>
<thead>
<tr>
<th></th>
<th>wohnt</th>
<th>hier</th>
<th>nicht</th>
</tr>
</thead>
<tbody>
<tr>
<td>john</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>does</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td></td>
<td>nicht</td>
</tr>
<tr>
<td>live</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>here</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Is the English word `does` aligned to the German `wohnt` (verb) or `nicht` (negation) or neither?
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 ⊆ P)

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

\[
AER(S, P; A) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}
\]

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

• However: different applications require different precision/recall trade-offs
symmetrization
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
Symmetrization

- Run IBM Model training in both directions
  → two sets of word alignment points

- Intersection: high precision alignment points

- Union: high recall alignment points

- Refinement methods explore the sets between intersection and union
Example

english to spanish

Maria no daba una bofetada a la bruja verde

Mary did not slap the green witch

spanish to english

Maria no daba una bofetada a la bruja verde

Mary did not slap the green witch

intersection
Growing Heuristics

### Add alignment points from union based on heuristics:

- directly/diagonally neighboring points
- finally, add alignments that connect unaligned words in source and/or target

### Popular method: grow-diag-final-and