Word-based models

Barry Haddow (slides by Philipp Koehn)

6th September 2011, Trento
Outline

- Lexical translation
- Alignment
- Expectation Maximization (EM) Algorithm
- IBM Word Translation Models
- Word Alignment as an Intermediate Step
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

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

- 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

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

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

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\text{das} & \text{Haus} & \text{ist} & \text{klein} \\
1 & 2 & 3 & 4 \\
\text{house} & \text{is} & \text{small} \\
a : \{1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4\}
\end{array}
\]
Inserting Words

- Words may be added during translation
  - The English \textit{just} does not have an equivalent in German
  - We still need to map it to something: special \textit{NULL} token

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{example_diagram.png}
\caption{Diagram showing word alignment and mapping}
\end{figure}

\[ a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 0, 5 \rightarrow 4\} \]
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 $\mathbf{f} = (f_1, ..., f_{l_f})$ 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
Example

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

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

\[
= 0.0028\epsilon
\]
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,
    \[ \rightarrow \text{we could estimate the parameters of our generative model} \]
  – if we had the parameters,
    \[ \rightarrow \text{we could estimate the alignments} \]
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 ...

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

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

• 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)
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

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

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

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

• 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 \( l_e = l_f = 2 \))

\[
\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} = \frac{\epsilon}{32} \prod_{j=1}^{2} t(e_j|f_{a(j)}) = \\
= t(e_1|f_0) t(e_2|f_0) + t(e_1|f_0) t(e_2|f_1) + t(e_1|f_0) t(e_2|f_2) + \\
+ t(e_1|f_1) t(e_2|f_0) + t(e_1|f_1) t(e_2|f_1) + t(e_1|f_1) t(e_2|f_2) + \\
+ t(e_1|f_2) t(e_2|f_0) + t(e_1|f_2) t(e_2|f_1) + t(e_1|f_2) t(e_2|f_2) = \\
= t(e_1|f_0) (t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) + \\
+ t(e_1|f_1) (t(e_2|f_1) + t(e_2|f_1) + t(e_2|f_2)) + \\
+ t(e_1|f_2) (t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2)) = \\
= (t(e_1|f_0) + t(e_1|f_1) + t(e_1|f_2)) (t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2))
\]
IBM Model 1 and EM: Expectation Step

- Combine what we have:

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

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

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

**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.     count\((e|f)\) = 0 **for all** \(e, f\)
5.     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-total\((e)\) = 0
10.        **for all** words \(f\) in \(f\) **do**
11.           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.         count\((e|f)\) += \(\frac{t(e|f)}{s\text{-total}(e)}\)
18.         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>$e$</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>
Assessing the Fit of the Model

- Use *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

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

• We would like to integrate a language model

• Bayes rule

\[
\arg\max_e p(e|f) = \arg\max_e \frac{p(f|e) \ p(e)}{p(f)} = \arg\max_e p(f|e) \ p(e)
\]
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
IBM Model 2

Adding a model of alignment

"natürlich ist das Haus klein"

of course the house is small

lexical translation step

alignment step

"of course is the house small"

Word-Based Models
IBM Model 3

Adding a model of fertility

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

**fertility step**

1. **ich**
2. **gehe**
3. **nicht**
4. **zum**
5. **haus**

**NULL insertion step**

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

**lexical translation step**

1. I
2. do
3. go
4. not
5. to
6. the house

**distortion step**

1. I
2. do
3. not go
4. to
5. the house
HMM Model

• Introduced after the IBM Models

• 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
Word Alignment in Practice

- IBM Models not used any more for translation.

- But still used for alignment
  - Important first step for training of most translation models
  - GIZA++ is best known implementatation

- Run sequence of IBM Models (e.g. m1, hmm, m3, m4)

- IBM Models limited to $n \rightarrow 1$, so run in both directions - then symmetrise
Symmetrizing Word Alignments

- Intersection of GIZA++ bidirectional alignments
- Start with intersection and heuristically add points from union

[Och and Ney, CompLing2003]
Symmetrisation Heuristics: grow-diag

- “grow-diag” heuristic iteratively adds neighbours
- Only adds \((e_i, f_j)\) if one of the words is unaligned
- In above, would add \((e_1, f_1)\)
- But not \((e_1, f_2)\), since both \(e_1\) and \(f_2\) have an alignment
Symmetrisation Heuristics: final-and

- “final-and” adds alignments for unaligned words
- Any alignment in union, where both words are unaligned
- In above example:
  - \((e_0, f_0)\) is added, since neither aligned
  - \((e_1, f_0)\) is not added, since \(f_0\) now aligned
  - \((e_1, f_1)\) added, since neither aligned
More Recent Work on Symmetricization

• 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

• Alignment by Agreement [Liang et al., 2006; Ganchev et al., 2008]
  – Train forward and backward simultaneously
  – Incorporate agreement directly into objective
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 classifier 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