Lecture 23:
Phrase-based MT (corrected)

Julia Hockenmaier
juliahmr@illinois.edu
3324 Siebel Center

Recap:
IBM models for MT

The IBM models

Use the noisy channel (Bayes rule) to get the best (most likely) target translation $e$ for source sentence $f$:

$$\arg \max_e P(e|f) = \arg \max_e P(f|e)P(e)$$

The translation model $P(f|e)$ requires alignments $a$:

$$P(f|e) = \sum_{a \in A(e,f)} P(a)P(f,a|e)$$

Generate $f$ and the alignment $a$ with $P(f,a|e)$:

$$P(f,a|e) = \prod_{j=1}^{m} P(e_j|a_j)P(a_j|f_{a(j-1)},f_{a(j-1)},e_j,m)P(f_j|a_{j-1},f_{a(j-1)},e_j,m)$$

$m = \#\text{words in } f_j$

Representing word alignments

<table>
<thead>
<tr>
<th>Foreign</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marie</td>
<td>a</td>
<td>traversé</td>
<td>le</td>
<td>lac</td>
<td>à</td>
<td>la</td>
<td>nage</td>
<td></td>
</tr>
<tr>
<td>Null</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mary</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>swam</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>across</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lake</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Every source word $f[i]$ is aligned to one target word $e[j]$ (incl. NULL). We represent alignments as a vector $a$ (of the same length as the source) with $a[i] = j$
IBM model 1: Generative process

For each target sentence $e = e_1...e_n$ of length $n$:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>Mary</td>
<td>swam</td>
<td>across</td>
<td>the</td>
<td>lake</td>
<td></td>
</tr>
</tbody>
</table>

1. Choose a length $m$ for the source sentence (e.g. $m = 8$)

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

2. Choose an alignment $a = a_1...a_m$ for the source sentence

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

3. Translate each target word $c_{e_j}$ into the source language

Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

| Translation | Marie | a traversé | le lac | à la | nage |

Expectation-Maximization (EM)

1. Initialize a first model, $M_0$

2. Expectation (E) step:
   Go through training data to gather expected counts $\langle \text{count}(lac, lake) \rangle$

3. Maximization (M) step:
   Use expected counts to compute a new model $M_{i+1}$
   $P_{i+1}(lac | lake) = \langle \text{count}(lac, lake) \rangle / \sum_w \text{count}(w, lake)$

4. Check for convergence:
   Compute log-likelihood of training data with $M_{i+1}$
   If the difference between new and old log-likelihood smaller than a threshold, stop. Else go to 2.

The E-step

Compute the expected count $\langle c(f, e|f, e) \rangle$:

$$\langle c(f, e|f, e) \rangle = \sum_{a \in A(f,e)} P(a|f,e) \cdot c(f,e|a,e,f)$$

$$P(a|f,e) = \frac{P(a,f|e)}{P(f|e)} = \frac{P(a,f|e)}{\sum_{a'} P(a',f|e)}$$

$$P(a,f|e) = \prod_j P(f_j|e_{a_j})$$

$$\langle c(f, e|f, e) \rangle = \sum_{a \in A(f,e)} \prod_j P(f_j|e_{a_j}) \cdot c(f,e|a,e,f)$$

Phrase-based translation models
Phrase-based translation models

Assumption: fundamental units of translation are phrases:

Phrase-based model of \( P(F \mid E) \):
1. Split target sentence deterministically into phrases \( e_1 \ldots e_n \).
2. Translate each target phrase \( e_p_i \) into source phrase \( f_p_i \) with translation probability \( \phi(f_p_i \mid e_p_i) \).
3. Reorder foreign phrases with distortion probability \( d(a_i-b_i-1) = c|a_i-b_i-1| \).

Translation probability \( P(f_p_i \mid e_p_i) \)

Phrase translation probabilities can be obtained from a phrase table:

<table>
<thead>
<tr>
<th>EP</th>
<th>FP</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>green witch</td>
<td>grüne Hexe</td>
<td>…</td>
</tr>
<tr>
<td>at home</td>
<td>zuhause</td>
<td>10534</td>
</tr>
<tr>
<td>at home</td>
<td>daheim</td>
<td>9890</td>
</tr>
<tr>
<td>is</td>
<td>ist</td>
<td>598012</td>
</tr>
<tr>
<td>this week</td>
<td>diese Woche</td>
<td>…</td>
</tr>
</tbody>
</table>

This requires phrase alignment

Phrase-based models of \( P(f \mid e) \)

Split target sentence \( e = e_1 \ldots e_n \) into phrases \( e_p_1 \ldots e_p_N \):

\[
\text{[The green witch] [is] [at home] [this week]}
\]

Translate each target phrase \( e_p_i \) into source phrase \( f_p_i \) with translation probability \( P(f_p_i \mid e_p_i) \):

\[
\text{[Diese Woche] [ist] [die grüne Hexe] [zuhause]}
\]

Arrange the set of source phrases \( \{ f_p_i \} \) to get \( s \) with distortion probability \( P(f_p \mid \{ f_p_i \}) \):

\[
P(f \mid e = \langle e_p_1, \ldots, e_p_l \rangle) = \prod_i P(f_p_i \mid e_p_i) P(f_p \mid \{ f_p_i \})
\]

Word alignment
### Phrase alignment

<table>
<thead>
<tr>
<th></th>
<th>Diese</th>
<th>Woche</th>
<th>ist</th>
<th>die</th>
<th>grüne</th>
<th>Hexe</th>
<th>zuhause</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>green</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>witch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>is</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>at</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>home</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>this</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>week</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Obtaining phrase alignments

We’ll skip over details, but here’s the basic idea:

For a given parallel corpus (F-E)
1. Train **two word aligners**, (F→E and E→F)
2. Take the **intersection** of these alignments to get a **high-precision** word alignment
3. **Grow** these high-precision alignments until all words in both sentences are included in the alignment.
   - Consider any pair of words in the **union** of the alignments, and incrementally add them to the existing alignments
4. Consider all phrases that are consistent with this improved word alignment

### Phrase-based models of \( P(f/e) \)

**Split target sentence** \( e=e_1..e_n \) into phrases \( ep_1..ep_N \):

\[ \text{[The green witch] [is] [at home] [this week]} \]

**Translate each target phrase** \( ep_i \) into source phrase \( fp_i \) with translation probability \( P(fp_i|ep_i) \):

\[ \text{[The green witch] = [die grüne Hexe], ...} \]

**Arrange the set of source phrases** \( \{fp_i\} \) to get \( s \) with distortion probability \( P(fp|\{fp_i\}) \):

\[ \text{[Diese Woche] [ist] [die grüne Hexe] [zuhause]} \]

\[
P(f|e = \langle ep_1, ..., ep_l \rangle = \prod_i P(fp_i|ep_i)P(fp|\{fp_i\})
\]
Translating

How do we translate a foreign sentence (e.g. “Diese Woche ist die grüne Hexe zuhause”) into English?
- We need to find \( \hat{e} = \arg \max_e P(f \mid e)P(e) \)
- There is an exponential number of candidate translations \( e \)
- But we can look up phrase translations \( e_p \) and \( P(f_p \mid e_p) \) in the phrase table:

<table>
<thead>
<tr>
<th>diese</th>
<th>Woche</th>
<th>ist</th>
<th>die</th>
<th>grüne</th>
<th>Hexe</th>
<th>zuhause</th>
</tr>
</thead>
<tbody>
<tr>
<td>this 0.2</td>
<td>week 0.7</td>
<td>is 0.8</td>
<td>the 0.3</td>
<td>green 0.3</td>
<td>witch 0.5</td>
<td>home 1.00</td>
</tr>
<tr>
<td>these 0.5</td>
<td>the green 0.4</td>
<td>sorceress 0.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>this week 0.6</td>
<td>green witch 0.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>is this week 0.4</td>
<td>the green witch 0.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Finding the best translation

How can we find the best translation efficiently?
There is an exponential number of possible translations.

We will use a heuristic search algorithm
We cannot guarantee to find the best (= highest-scoring) translation, but we’re likely to get close.

We will use a “stack-based” decoder
(If you’ve taken Intro to AI: this is A* (“A-star”) search)
We will score partial translations based on how good we expect the corresponding complete translation to be.
Or, rather: we will score partial translations on how bad we expect the corresponding complete translation to be.
That is, our scores will be costs (high=bad, low=good)

Generating a (random) translation

1. Pick the first Target phrase \( e_{p1} \) from the candidate list.
   \[ P := P_{LM}(\text{<s>} \mid e_{p1})P_{Trans}(f_{p1} \mid e_{p1}) \]
   \[ E = \text{the}, F = \text{<....die...> } \]

2. Pick the next target phrase \( e_{p2} \) from the candidate list
   \[ P := P \times P_{LM}(e_{p2} \mid e_{p1})P_{Trans}(f_{p2} \mid e_{p2}) \]
   \[ E = \text{the green witch}, F = \text{<....die grüne Hexe...> } \]

3. Keep going: pick target phrases \( e_{pi} \) until the entire source sentence is translated
   \[ P := P \times P_{LM}(e_{pi} \mid e_{p1...i-1})P_{Trans}(f_{pi} \mid e_{pi}) \]
   \[ E = \text{the green witch is}, F = \text{<....ist die grüne Hexe...> } \]

Scoring partial translations

Assign expected costs to partial translations \((E,F)\):

\[ \text{expected\_cost}(E,F) = \text{current\_cost}(E,F) + \text{future\_cost}(E,F) \]

The current cost is based on the score of the partial translation \((E,F)\)
\[ \text{e.g. current\_cost}(E,F) = \log P(E)P(F \mid E) \]

The (estimated) future cost is a lower bound on the actual cost of completing the partial translation \((E,F)\):

\[ \text{true\_cost}(E,F) \geq \text{expected\_cost}(E,F) \geq \text{actual\_future\_cost}(E,F) \]

(The estimated future cost ignores the distortion cost)
Stack-based decoding

Maintain a priority queue (=‘stack’) of partial translations (hypotheses) with their expected costs. Each element on the stack is open (we haven’t yet pursued this hypothesis) or closed (we have already pursued this hypothesis).

At each step:
- **Expand** the best open hypothesis (the open translation with the lowest expected cost) in all possible ways.
- These new translations become new open elements on the stack.
- **Close** the best open hypothesis.

Additional Pruning (n-best / beam search):
Only keep the n best open hypotheses around.

We’re done with this node now (all continuations have a lower cost).

Expand one of these new yellow nodes next.
Stack-based decoding

Expand the yellow node with the lowest cost

CS447: Natural Language Processing (J. Hockenmaier)

Cost: 999

Stack-based decoding

Expand the next node with the lowest cost

CS447: Natural Language Processing (J. Hockenmaier)

Cost: 999

We always expand the best (lowest-cost) node, even if it's not the last one introduced

CS447: Natural Language Processing (J. Hockenmaier)
MT evaluation

Automatic evaluation: BLEU

Evaluate candidate translations against several reference translations.

C1: It is a guide to action which ensures that the military always obeys the commands of the party.
C2: It is to insure the troops forever hearing the activity guidebook that party direct
R1: It is a guide to action that ensures that the military will forever heed Party commands.
R2: It is the guiding principle which guarantees the military forces always being under the command of the Party.
R3: It is the practical guide for the army always to heed the directions of the party.

The **BLEU score** is based on **N-gram precision**: How many n-grams in the candidate translation occur also in one of the reference translation?

BLEU details

For $n \in \{1,...,4\}$, compute the (modified) precision of all $n$-grams:

$$
Prec_n = \frac{\sum_{c \in C} \sum_{n\text{-gram} \in c} \text{MaxFreq}_{\text{ref}}(n\text{-gram})}{\sum_{c \in C} \sum_{n\text{-gram} \in c} \text{Freq}_c(n\text{-gram})}
$$

$\text{MaxFreq}_{\text{ref}}(\text{the party}) =$ max. count of ‘the party’ in one reference translation.

$\text{Freq}_c(\text{the party}) =$ count of ‘the party’ in candidate translation $c$.

Penalize short candidate translations by a **brevity penalty** $BP$

$c =$ length (number of words) of the whole candidate translation corpus

$r =$ Pick for each candidate the reference translation that is closest in length; sum up these lengths.

**Brevity penalty** $BP = \exp(1-c/r)$ for $c \leq r$; $BP = 1$ for $c>r$

(BP ranges from 0 for $c=0$ to 1 for $c=r$)

BLEU score

The BLEU score is the geometric mean of the precision of the unigrams, bigrams, trigrams, quadrigrams, weighted by the brevity penalty $BP$.

$$
\text{BLEU} = BP \times \exp \left( \frac{1}{N} \sum_{n=1}^{N} \log Prec_n \right)
$$
Human evaluation

We want to know whether the translation is “good” English, and whether it is an accurate translation of the original.

- Ask human raters to judge the fluency and the adequacy of the translation (e.g. on a scale of 1 to 5)
- Correlated with fluency is accuracy on cloze task:
  Give rater the sentence with one word replaced by blank. Ask rater to guess the missing word in the blank.
- Similar to adequacy is informativeness
  Can you use the translation to perform some task (e.g. answer multiple-choice questions about the text)

Machine translation models

Current MT models all rely on statistics.

Many current models do estimate $P(E \mid F)$ directly, but may use features based on language models (capturing $P(E)$) and IBM-style translation models ($P(F \mid E)$) internally.

There are a number of syntax-based models, e.g. using synchronous context-free grammars, which consist of pairs of rules for the two languages in which each RHS NT in language A corresponds to a RHS NT in language B:

Language A: $XP \rightarrow YP ZP$  Language B: $XP \rightarrow ZP YP$

More recent developments

Neural network-based approaches:
  Recurrent neural networks (RNN) can model sequences (e.g. strings, sentences, etc.)
  Use one RNN (the encoder) to process the input in the source language
  Pass its output to another RNN (the decoder) to generate the output in the target language

See e.g. [http://www.tensorflow.org/tutorials/seq2seq/index.md#sequence-to-sequence_basics]