The IBM models

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

\[ \text{arg max}_e P(e|f) = \text{arg max}_e P(f|e)P(e) \]

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

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

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

\[ P(f,a|e) = \frac{P(m|e)}{\text{Length: } |f|=m} \prod_{j=1}^{m} P(a_j|a_{j-1},f_{1:j-1},m,e)P(f_j|a_{1:j-1},f_{1:j-1},e,m) \]

The IBM alignment models

Model parameters

Length probability \( P(m \mid n) \):
What's the probability of generating a source sentence of length \( m \) given a target sentence of length \( n \)?
Count in training data

Alignment probability: \( P(a \mid m, n) \):
Model 1 assumes all alignments have the same probability:
For each position \( a_1 \ldots a_m \), pick one of the \( n+1 \) target positions uniformly at random

Translation probability: \( P(f_i = lac \mid a_i = i, e_i = lake) \):
In Model 1, these are the only parameters we have to learn.
Representing word alignments

<table>
<thead>
<tr>
<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 traver sé</td>
<td>le lac</td>
<td>à la nage</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>NULL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Mary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>swim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>across</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>the</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>lake</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: details

The **length probability** is constant: $P(m|e) = \epsilon$

The **alignment probability** is uniform ($n =$ length of target string): $P(a_i|e) = 1/(n+1)$

The **translation probability** depends only on $e_{ai}$ (the corresponding target word): $P(f_j|e_{ai})$

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

$$P(f,a|e) = \epsilon \prod_{j=1}^{m} \frac{1}{n+1} P(f_j|e_{ai})$$

$$P(f,a|e) = \frac{\epsilon}{(n+1)^m} \prod_{j=1}^{m} P(f_j|e_{ai})$$

IBM model 1: Generative process

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

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

<table>
<thead>
<tr>
<th>0</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>NULL</td>
<td>Mary</td>
<td>swim</td>
<td>across</td>
<td>the</td>
<td>lake</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

   Each $a_i$ corresponds to a word $e_i$ in $e$: $0 \leq a_i \leq n$

<table>
<thead>
<tr>
<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>Position</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Alignment</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

3. **Translate each target word $e_{ai}$ into the source language**

<table>
<thead>
<tr>
<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>Position</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Alignment</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Translation</td>
<td>Marie</td>
<td>a traver sé</td>
<td>le lac</td>
<td>à la nage</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Finding the best alignment

How do we find the **best alignment** between $e$ and $f$?

$$\hat{a} = \arg \max_a P(f,a|e)$$

$$\hat{a} = \arg \max_a \epsilon \prod_{j=1}^{m} P(f_j|e_{ai})$$

$$\hat{a}_j = \arg \max_{a_j} P(f_j|e_{ai})$$
Learning translation probabilities

The only parameters that need to be learned are the translation probabilities \( P(f \mid e) \)

\[
P(f_j = lac \mid e_i = lake)
\]

If the training corpus had word alignments, we could simply count how often ‘lake’ is aligned to ‘lac’:

\[
P( lac \mid lake) = \frac{\text{count}(lac, lake)}{\sum_w \text{count}(w, lake)}
\]

But we don’t have word alignments. So, instead of relative frequencies, we have to use expected relative frequencies:

\[
P( lac \mid lake) = \frac{\langle \text{count}(lac, lake) \rangle}{\sum_w \text{count}(w, lake)}
\]

Training Model 1 with EM

The only parameters that need to be learned are the translation probabilities \( P(f \mid e) \)

We use the EM algorithm to estimate these parameters from a corpus with \( S \) sentence pairs \( s = (f^{(s)}, e^{(s)}) \) with alignments \( A(f^{(s)}, e^{(s)}) \)

- **Initialization**: guess \( P(f \mid e) \)
- **Expectation step**: compute expected counts

\[
\langle c(f, e) \rangle = \sum_{s \in S} \langle c(f, e|s^{(s)}, f^{(s)}) \rangle
\]

- **Maximization step**: recompute probabilities \( P(f \mid e) \)

\[
P(f \mid e) = \frac{\langle c(f, e) \rangle}{\sum_f \langle c(f', e) \rangle}
\]

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 \( M_{i+1} \)

\[
P_{i+1}( lac \mid lake) = \frac{\langle \text{count}(lac, lake) \rangle}{\sum_w \text{count}(w, lake)}
\]

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 \( \langle c(f, e|f, e) \rangle \):

\[
\langle c(f, e|f, e) \rangle = \sum_{a \in A(f, e)} \frac{P(a|f, e) \cdot c(f, e|a, e, f)}{\sum_{a'} P(a'|f, e)}
\]

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

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

\[
\langle c(f, e|f, e) \rangle = \sum_{a \in A(f, e)} \frac{\prod_j P(f_j|e_{a_j}) \cdot c(f, e|a, e, f)}{\sum_{a'} \prod_j P(f_j|e_{a'_j})}
\]
Other translation models

Model 1 is a very simple (and not very good) translation model.

IBM models 2-5 are more complex. They take into account:
- “fertility”: the number of foreign words generated by each target word
- the word order and string position of the aligned words

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 $ep_1...ep_n$
2. Translate each target phrase $ep_i$ into source phrase $fp_i$
   with translation probability $\varphi(fp_i \mid ep_i)$
3. Reorder foreign phrases with distortion probability
   $d(a_i - b_{i-1}) = c|a_i - b_{i-1} - 1|$
   $a_i$ = start position of the source phrase generated by the current (i-th) target phrase, $ep_i$
   $b_{i-1}$ = end position of the source phrase generated by the preceding (i-1-th) target phrase, $ep_{i-1}$
   $|a_i - b_{i-1} - 1| = \#words in source sentence between translations of $ep_i$ and $ep_{i-1}$

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

Split target sentence $e = e_1...e_n$ into phrases $ep_1...ep_N$:

[The green witch] [is] [at home] [this week]

Translate each target phrase $ep_i$ into source phrase $fp_i$
with phrase translation probability $\varphi(fp_i \mid ep_i)$:

[The green witch] = [die grüne Hexe], ...

Arrange the set of source phrases $\{fp_i\}$ to get $f$
with distortion probability $d(a_i - b_{i-1})$

[Diese Woche] [ist] [die grüne Hexe] [zuhause]

NB: we can redefine $P_{\text{trans}}(fp_i \mid ep_i)$ as

$P_{\text{trans}}(fp_i \mid ep_i) = \varphi(fp_i \mid ep_i) \cdot d(a_i - b_{i-1})$
Translation probability $\phi(fp_i | ep_i)$

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

Word alignment

<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>The</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>
</tr>
<tr>
<td>witch</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>
</tr>
<tr>
<td>at</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>
</tr>
<tr>
<td>this</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>
</tr>
</tbody>
</table>

Phrase alignment

<table>
<thead>
<tr>
<th>Deze</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>
</tr>
<tr>
<td>green</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>
</tr>
<tr>
<td>is</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>
</tr>
<tr>
<td>home</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>
</tr>
<tr>
<td>week</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Decoding
(for phrase-based MT)
Translating

How do we translate a foreign sentence (e.g. “Diese Woche ist die grüne Hexe zuhause”) into English?
- We need to find \( \hat{e} = \text{argmax}_e P(f \mid e) P(e) \)
- There is an exponential number of candidate translations \( e \)
- But we can look up phrase translations \( e_p \) and \( \phi(f_p \mid e_p) \) 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></td>
<td>the green 0.4</td>
<td>sorceress 0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>this week 0.6</td>
<td></td>
<td>green witch 0.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>is this week 0.4</td>
<td></td>
<td>the green witch 0.7</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 completed 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.
   \( P := P_{LM}(<s> \mid e_{p1}) P_{Trans}(f_{p1} \mid e_{p1}) \)
   \( E = \text{the}, F = \langle \ldots \text{die} \ldots \rangle \)

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

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

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 \((E, F)\)

The (estimated) future cost is an upper bound on the actual cost of completing the partial translation \((E, F)\):
\[
\text{true\_cost}(E, F) = \text{current\_cost}(E, F) + \text{actual\_future\_cost}(E, F) \leq \text{expected\_cost}(E, F) = \text{current\_cost}(E, F) + \text{future\_cost}(E, F)
\]

because \( \text{actual\_future\_cost}(E, F) \leq \text{future\_cost}(E, F) \)
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

MT evaluation

Evaluate candidate translations against several reference translations.

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

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

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

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

Penalize short candidate translations by a brevity penalty \( BP \)

\( c = \text{length (number of words) of the whole candidate translation corpus} \)
\( r = \text{Pick for each candidate the reference translation that is closest in length; sum up these lengths.} \)

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

(\( BP \) ranges from \( e \) 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)

Summary:

Machine Translation
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: $XP \rightarrow ZP YP$

Very 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

Today’s key concepts

Why is machine translation hard?
Linguistic divergences: morphology, syntax, semantics

Different approaches to machine translation:
Vauquois triangle
Statistical MT: Noisy Channel, IBM Model 1 (more on this next time)