IBM Model 1 for Machine Translation

Micha Elsner

March 28, 2014
Machine translation

A key area of computational linguistics

- Bar-Hillel points out that human-like translation requires understanding of the text
- For instance to disambiguate words
- As with generation, MT research follows two paths
  - Shallow text-to-text methods
  - Deeper methods using “interlingua”/semantics
Shallow methods

We’ll study an older method from the “shallow” word-based school

- Still used as an ingredient in more sophisticated translations
- Bears some similarity to lost-language decipherment methods
  - Ex. Rosetta Stone, Linear B...
Notation

Given:

- English sentence $E_1:n$
- French (or “foreign”) sentence $F_1:m$
  - The letters are conventional regardless of the actual languages involved
- We want to model $P(E|F)$, and then search for:

$$\max_{E} P(E|F)$$

We’ll assume we have bilingual parallel text for training:
$D : (E^i, F^i)$
- Same documents in English and French
- Canadian Hansards
The noisy channel

- We’ve seen this before (discussing speech recognition)
- Studied in information theory (Shannon)...
- But really just a specific use of Bayes’ rule

\[
\max_E P(E|F) = \max_E P(E)P(F|E)
\]

- Dropping the constant \(P(F)\) in the denominator
- \(P(E)\) is the “source” model— language model for English
- \(P(F|E)\) is the “channel” or “noise” model

When I look at an article in Russian, I say: ’This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode. — Warren Weaver
Why noisy channels?

\[
\max_E P(E|F) = \max_E P(E)P(F|E)
\]

▶ Seems like we haven’t made any progress
▶ But: we’re good at language modeling
  ▶ Also, can use monolingual data
▶ The LM can take over enforcing grammaticality in English
▶ Translation model \( P(F|E) \) doesn’t have to know as much grammar...
▶ Mainly focused on content
The channel model

Need to build a model of $P(F|E)$

- Basic intuition: model is an En-Fr dictionary
- Translation: pick an En word, look up and take a Fr word
  - This word-to-translation correspondence is called *alignment*
- This *alignment* idea is what makes it an IBM model
- Add to the sentence
  - We don’t know any syntax, so just throw it in wherever
- Result: each Fr word corresponds to a *single English word*
  - Also a NULL English word to model “things French people just say”

Brown+al `90

<table>
<thead>
<tr>
<th>En</th>
<th>He</th>
<th>is going by</th>
<th>train</th>
<th>NULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ger:</td>
<td>Er</td>
<td>fährt</td>
<td>mit</td>
<td>dem</td>
</tr>
</tbody>
</table>
Bad cases for alignment

I know you are a cat

Je sais que tu es un chat

the long distance train station

das Fernbahnhof
Formalization

Introduce $A_{1:m}$, an alignment variable for each French word

- $a_i \in [0 \ldots n]$ is a pointer to the English word which is the source of this French word

Probability of French word depends on a single English word (selected by $a_i$):

$$P(F_i|E_{1:n}, a_i) = P_{tr}(F_i|E_{a_i})$$

$P_{tr}$ is the dictionary:

<table>
<thead>
<tr>
<th>French</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>le</td>
<td>0.52</td>
</tr>
<tr>
<td>de</td>
<td>0.162</td>
</tr>
<tr>
<td>la</td>
<td>0.151</td>
</tr>
<tr>
<td>les</td>
<td>0.107</td>
</tr>
<tr>
<td>sais</td>
<td>0.267</td>
</tr>
<tr>
<td>que</td>
<td>0.254</td>
</tr>
<tr>
<td>savons</td>
<td>0.112</td>
</tr>
<tr>
<td>savoir</td>
<td>0.11</td>
</tr>
</tbody>
</table>
Probability of all the French words

A single French word:

\[ P(F_i \mid E_{1:n}, A_i) = P_{tr}(F_i \mid E_{a_i}) \]

All French words are independent given \( E, A \):

\[ P(F_{1:m} \mid E_{1:n}, A_{1:m}) = \prod_{i=1}^{m} P_{tr}(F_i \mid E_{1:n}, A_i) \]

This would be all we need... if we knew \( A \)

- One can imagine paying students to annotate it
- But this would be slow and expensive
- Instead we will use unsupervised learning
Setting up an EM algorithm

Probability of a sentence:

\[ P(F_{1:m} | E_{1:n}, A_{1:m}) = \prod_{i=1}^{m} P_{tr}(F_i | E_{1:n}, A_i) \]

- In hard EM, would alternate between predicting \( A \) and learning \( P_{tr} \)
- However, soft EM is known to work well here
  - For more complex IBM models, hard EM is not as good (Och and Ney 97)
- Will calculate *probability* of various \( A \) and expected counts

Maximizing the probability of the sentence:

- Marginalizing over \( A \)

\[ P(F_{1:m} | E_{1:n}) = \sum_A \left( \prod_{i=1}^{m} P_{tr}(F_i | E_{1:n}, A_i) \right) P(A_{1:m}) \]
Computing the responsibilities

Soft EM: need partial counts to estimate $P_{tr}$

$$\hat{P}_{tr}(f|e) = \frac{\#(F_i = f, a_i = j, E_j = e)}{\#(E_j = e)}$$

Need to know probability French word $f$ is aligned to English word $e$ (applying Bayes’ rule)

$$P(A_i = j|F_{1:m}, E_{1:n}) = \frac{P(F_{1:m}|A_i = j, E_{1:n})P(A_i = j)}{P(F_{1:m})}$$

- To do this, will need to know more about $P(A)$
Probabilities of alignment

What is $P(A)$?

- Tells us where in English sentence we should look for the source of each French word
- This depends on syntax
- Fr, En both SVO, so go in order
- German is SOV so look for verbs at the end
- In IBM model 1, make simplest possible assumption
  - Look anywhere! (uniform)

**IBM model 1 assumption**

$$P(A_{1:m}|E_{1:n}) = \prod_i P(A_i|E_{1:n}) = \prod_i \frac{1}{n}$$
Simplifying

We can use the assumption to simplify our objective

- So we can compute responsibilities for EM

\[
P(A_{1:m} | E_{1:n}) = \prod_i P(A_i | E_{1:n}) = \prod_i \frac{1}{n}
\]

\[
P(F_{1:m} | E_{1:n}) = \sum_A \left( \prod_{i=1}^{m} P_{tr}(F_i | E_{1:n}, A_i) P(A_{1:m}) \right)
\]

\[
P(F_{1:m} | E_{1:n}) = \sum_A \left( \prod_{i=1}^{m} P_{tr}(F_i | E_{1:n}, A_i) \right) \prod_i \frac{1}{n}
\]

\[
P(F_{1:m} | E_{1:n}) = \sum_A \left( \prod_{i=1}^{m} P_{tr}(F_i | E_{1:n}, A_i) \frac{1}{n} \right)
\]

\[
P(F_{1:m} | E_{1:n}) = \prod_{i=1}^{m} \left( \sum_{A_i \in [1...n]} P_{tr}(F_i | E_{1:n}, A_i) \frac{1}{n} \right)
\]
Computing the responsibilities

\[ P(F_{1:m}|E_{1:n}) = \prod_{i=1}^{m} \left( \sum_{A_i \in [1...n]} P_{tr}(F_i|E_{1:n}, A_i) \frac{1}{n} \right) \]

For each French word, we have:

\[ P(F_i|E_{1:n}) = \sum_{A_i \in [1...n]} P_{tr}(F_i|E_{1:n}, A_i) \frac{1}{n} \]

We need:

\[ P(A_i = j|F_{1:m}, E_{1:n}) = \frac{P(F_{1:m}|A_i = j, E_{1:n})P(A_i = j)}{P(F_{1:m})} \]

Independence assumptions: drop dependence on all French words but \( F_i \):

\[ P(A_i = j|F_i, E_{1:n}) = \frac{P(F_i|A_i = j, E_{1:n})P(A_i = j)}{P(F_i)} \]
Continued...

\[ P(A_i = j | F_i, E_{1:n}) = \frac{P(F_i | A_i = j, E_{1:n})P(A_i = j)}{P(F_i)} \]

Numerator: use fact that \( A_i = j \) and \( P(A) \) is uniform;

\[ P(A_i = j | F_i, E_{1:n}) = \frac{P_{tr}(F_i | E_j)^1}{P(F_i)} \]

Denominator: derived on prev slide (or sum over numerators):

\[ \sum_{A_i \in [1...n]} P_{tr}(F_i | E_{1:n}, A_i)^1 \]

Thus (note that the \( \frac{1}{n} \) divides out):

\[ P(A_i = j | F_i, E_{1:n}) = \frac{P_{tr}(F_i | E_j)^1}{\sum_{A_i \in [1...n]} P_{tr}(F_i | E_{1:n}, A_i)^1} \]
Example

Running EM: similar to Rosetta Stone decipherment problems:

<table>
<thead>
<tr>
<th>German</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ich habe eine Katze</td>
<td>I have a cat</td>
</tr>
<tr>
<td>Ich habe eine Matte</td>
<td>I have a mat</td>
</tr>
<tr>
<td>Die Katze sitz auf der Matte</td>
<td>The cat sits on the mat</td>
</tr>
</tbody>
</table>

Begin with $P_{tr}$ uniform:

- In E-step 1, calculate $P(A_i = j)$ for all German words
- Since we know nothing, these are uniform over English words in corresponding sentence
- In ss 1, Katze is either I, have, a, cat or NULL (each $P = \frac{1}{5}$)

Counts:

<table>
<thead>
<tr>
<th>Counts</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#(I, Katze)</td>
<td>1/5 (from ss 1)</td>
</tr>
<tr>
<td>#(have, Katze)</td>
<td>1/5 (from ss 1)</td>
</tr>
<tr>
<td>#(sits, Katze)</td>
<td>1/7 (from ss 3)</td>
</tr>
<tr>
<td>#(cat, Katze)</td>
<td>1/7 + 1/5 (from ss 1 and 3)</td>
</tr>
<tr>
<td>#(the, Katze)</td>
<td>2/7 (from ss 3)</td>
</tr>
</tbody>
</table>
M-step

We re-estimate:

\[ \hat{P}_{tr}(Katze|cat) = \frac{1/7 + 1/5}{\text{various counts for things aligned to cat}} \]

- Alignments to *cat* will probably increase
- Alignments of other words to *cat* will decrease
  - Since we now suspect *cat* translates as *Katze*
- This will help us deal with other words
  - *habe* less likely to be *cat*...
  - So more likely to be *have*
Conclusion

- IBM: shallow (word-to-word) translation
- Based on noisy channel (LM and channel model)
- Channel model based on word-to-word alignments and dictionary
- IBM 1: alignments are uniform (no syntax)
- Use EM to learn alignments/dictionary