Phrase-Based MT

Ondřej Bojar

March 26, 2020
Outline

• PBMT Overview.
• Reminder: Log-linear model.
• PBMT Model.
  • Features Used.
  • Traditional PBMT “Pipeline”
• Translating with PBMT (Decoding)
  • Translation Options and Stack-Based Beam Search
  • MT is NP-Hard.
  • Pruning, Future Cost Estimation.
  • Local and Non-Local Features.
• Minimum Error-Rate Training.
• Moses as the implementation.
Overview: Phrase-Based MT

Phrase-based MT: choose such segmentation of input string and such phrase “replacements” to make the output sequence “coherent” (3-grams most probable).

This time around = Nyní
they ’re moving = zareagovaly
even = dokonce ještě
... = ...

This time around, they ’re moving even faster = Dokonce ještě rychleji
... = ...

Nyní, zareagovaly, dokonce, ještě, rychleji.

faster
even
dokonce jaště
around
This
Nyní
zareagovaly
jestě
rychleji.

moving
’re
they
around
time
This

This time around = Nyní
they ’re moving = zareagovaly
even = dokonce ještě
... = ...

This time around, they ’re moving even faster = Dokonce ještě rychleji
... = ...

Nyní, zareagovaly, dokonce, ještě, rychleji.
Reminder: Log-Linear Model

• $p(e_1^I|f_1^J)$ is modelled as a weighted combination of models, called “feature functions”: $h_1(\cdot, \cdot) \ldots h_M(\cdot, \cdot)$

$$p(e_1^I|f_1^J) = \exp\left(\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J)\right) \sum_{e_1^{I'}} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(e_1^{I'}, f_1^J)\right)$$  \hspace{1cm} (1)

• The constant denominator not needed in maximization:

$$\hat{e}_1^I = \arg\max_{I, e_1^I} \frac{\exp\left(\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J)\right)}{\sum_{e_1^{I'}} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(e_1^{I'}, f_1^J)\right)}$$ \hspace{1cm} (2)

Model weights $\lambda_m^M$ specify the relative importance of features.
Phrase-Based Translation Model

• Captures the basic assumption of phrase-based MT:
  1. Segment source sentence $f^J_1$ into $K$ phrases $\tilde{f}_1, \ldots, \tilde{f}_K$.
  2. Translate each phrase independently: $\tilde{f}_k \rightarrow \tilde{e}_k$.
  3. Concatenate translated phrases (with possible reordering $R$):
     $\tilde{e}_{R(1)} \ldots \tilde{e}_{R(K)}$

• In theory, the segmentation $s^K_1$ is a hidden variable in the maximization, we should be summing over all segmentations: (Note the three args in $h_m(\cdot, \cdot, \cdot)$ now.)

$$\hat{e}^I_1 = \argmax_{I, e_1} \sum_{s^K_1} \exp(\sum_{m=1}^{M} \lambda_m h_m(e^I_1, f^J_1, s^K_1))$$

(3)

• In practice, the sum is approximated with a max (the biggest element only):

$$\hat{e}^I_1 = \argmax_{I, e_1} \max_{s^K_1} \exp(\sum_{m=1}^{M} \lambda_m h_m(e^I_1, f^J_1, s^K_1))$$

(4)
Commonly Used Features of PBMT

- **Phrase translation probability:**
  \[ h_{\text{Phr}}(f^J_1, e^I_1, s^K_1) = \log \prod_{k=1}^{K} p(\tilde{f}_k | \tilde{e}_k) \]
  where \( p(\tilde{f}_k | \tilde{e}_k) = \frac{\text{count}(\tilde{f}, \tilde{e})}{\text{count}(\tilde{e})} \)
  ⇒ Are all used units \( \tilde{f} \leftrightarrow \tilde{e} \) likely translations?

- **Word count/penalty:** \( h_{\text{wp}}(e^I_1, \cdot, \cdot) = I \)
  ⇒ Do we prefer longer or shorter output?

- **Phrase count/penalty:** \( h_{\text{pp}}(\cdot, \cdot, s^K_1) = K \)
  ⇒ Do we prefer translation in more or fewer less-dependent bits?

- **Reordering model:** different basic strategies (Lopez, 2009)
  ⇒ Which source spans can provide continuation at a moment?

- **\( n \)-gram LM:** \( h_{\text{LM}}(\cdot, e^I_1, \cdot) = \log \prod_{i=1}^{I} p(e_i | e_{i-n+1}^{i-1}) \)
  ⇒ Is output \( n \)-gram-wise coherent?
“Training the Translation Model”

1. Find relevant parallel texts.
2. Align at the level of sentences.
3. Align at the level of words.
4. Extract translation units, with scores (co-oc. stats.). (Language Model similar, “simple” words co-oc. stats, no alignment.)

“Tuning” (“MERT”) = Actual training in the ML sense

5. Identify TM/LM/other model component weights.

Translation: = Inference in the ML sense

6. Decompose input into known units.
7. Search for best combinations of units.
Estimating Phrase Translation Probs

The most important feature: phrase-to-phrase translation:

\[
h_{\text{Phr}}(f_1^J, e_1^I, s_1^K) = \log \prod_{k=1}^{K} p(\tilde{f}_k | \tilde{e}_k) \quad (5)
\]

The conditional probability of phrase \(\tilde{f}_k\) given phrase \(\tilde{e}_k\) is estimated from relative frequencies:

\[
p(\tilde{f}_k | \tilde{e}_k) = \frac{\text{count}(\tilde{f}, \tilde{e})}{\text{count}(\tilde{e})} \quad (6)
\]

- \(\text{count}(\tilde{f}, \tilde{e})\) is the number of co-occurrences of a phrase pair \((\tilde{f}, \tilde{e})\) that are consistent with the word alignment
- \(\text{count}(\tilde{e})\) is the number of occurrences of the target phrase \(\tilde{e}\) in the training corpus.
- \(h_{\text{Phr}}\) is usually used twice, in both directions: \(p(\tilde{f}_k | \tilde{e}_k)\) and \(p(\tilde{e}_k | \tilde{f}_k)\)
“Training” = Phrase Extraction

This time around, they’re moving even faster...

Extract all phrases (up to max-phrase-len) consistent with the word alignment.

• Long and short.
• Overlapping in all ways.

Score them (Eq. 6) ⇒ Phrase Table
Phrase Table in Moses

Given parallel training corpus, phrases are extracted and scored:

<table>
<thead>
<tr>
<th>Phrase</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>in europa</td>
<td>0.829007</td>
<td>0.207955</td>
<td>0.801493</td>
<td>0.492402</td>
<td>2.718</td>
</tr>
<tr>
<td>europas</td>
<td>0.0251019</td>
<td>0.066211</td>
<td>0.0342506</td>
<td>0.0079563</td>
<td>2.718</td>
</tr>
<tr>
<td>in eu</td>
<td>0.018451</td>
<td>0.00100126</td>
<td>0.0319584</td>
<td>0.0196869</td>
<td>2.718</td>
</tr>
</tbody>
</table>

The scores are: \( \phi(\cdot) = \log p(\cdot) \)

- phrase translation probabilities: \( \phi_{\text{phr}}(f|e) \) and \( \phi_{\text{phr}}(e|f) \)
- lexical weighting: \( \phi_{\text{lex}}(f|e) \) and \( \phi_{\text{lex}}(e|f) \) (Koehn, 2003)

\[
\phi_{\text{lex}}(f|e) = \log \max_{a \in \text{alignments of } (f,e)} \left\{ \prod_{i=1}^{\|f\|} \frac{1}{|\{j|(i,j) \in a\|} \sum_{\forall (i,j) \in a} p(f_i|e_j) \right\} \tag{7}
\]

- phrase penalty (always \( e^1 = 2.718 \))
Translation with phrase-based model has two main stages:

1. Translation Options Preparation.
   - Search the phrase table for all phrases applicable to the input sentence.

2. Decoding (Main Search).
   - Gradual hypothesis expansion.
   - Output produced left-to-right.
   - Input consumed in any order.
er geht ja nicht nach hause

it, it, he is goes go yes, of course not does not is not after to according to in home under house return home do not

is are is after all does not not is not are not is not a
Stage 2: Decoding (Beam Search)

consult phrase translation table for all input phrases
Stage 2: Decoding (Beam Search)

initial hypothesis: no input words covered, no output produced
Stage 2: Decoding (Beam Search)

pick any translation option, create new hypothesis
Stage 2: Decoding (Beam Search)

create hypotheses for all other translation options
Stage 2: Decoding (Beam Search)

er geht ja nicht nach hause

also create hypotheses from created partial hypothesis
Stage 2: Decoding (Beam Search)

er geht ja nicht nach hause
are
it
he
goes
does not
yes
go
to
home
home
backtrack from highest scoring complete hypothesis
Interlude: MT is NP-Hard (1/2)

- Translation options lead to exponentially many hypotheses.
- Indeed, MT is NP-hard for at least two reasons:
  - Finding the best word ordering.
  - Covering with multi-word units.
- Remember the NP-hardness proof strategy:
  - Use MT as a black box to solve an NP-complete task.

With a 2-gram language model, finding the best word ordering solves the Hamilton Circuit or Travelling Salesman Problem. (Knight, 1999)
Interlude: MT is NP-Hard (2/2)

Selecting a set of multi-word translations to cover the whole sentence solves Minimum Set Cover Problem. (Knight, 1999)

The sentence is:
(However, she cooked and left.)
Fighting the Complexity

See slides 17–32 by Barry Haddow.

- Hypothesis Recombination.
- Stack-based Pruning.
- Future Cost Estimation.
Local and Non-Local Features

- Local features decompose along hypothesis construction.
  - Phrase- and word-based features.
- Non-local features span the boundaries (e.g. LM).
Minimum Error Rate Training (Och, 2003)
Effects of Weights

- Higher phrase penalty chops sentence into more segments.
- Too strong LM weight leads to words dropped.
- Negative LM weight leads to obscure wordings.
Moses Decoder

- http://www.statmt.org/moses
- Moses Installation Tutorial.
Phrase-based MT:  
• is a log-linear model  
• decomposes sentence into contiguous phrases (\(\text{MTU}\))  
• assumes phrases relatively independent of each other  
• search has two parts:  
  • lookup of all relevant translation options  
  • stack-based beam search, gradually expanding hypotheses

To train a PBMT system:  
1. Align words.  
2. Extract (and score) phrases consistent with word alignment.  
3. Optimize weights (MERT).

Best implementation: Moses Decoder.
References


