Log-linear Models for Word Alignment

Yang Liu, Qun Liu and Shouxun Lin

Institute of Computing Technology
Chinese Academy of Sciences
Categories of Alignment
Approaches

• Statistical approaches
  – based on well-founded probabilistic models
  – depend on unknown parameters that are learned from training data

• Heuristic approaches
  – use various similarity functions between the types of two languages
Previous Work

- Combination of association clues (Tiedemann, 2003)
- Model 6, a log-linear combination of IBM Model 4 and HMM model (Och and Ney, 2003)
- A probability model, allowing easy integration of context-specific features (Cherry and Lin, 2003)
Log-linear models, which are very suitable to incorporate additional dependencies, have been successfully applied to statistical machine translation (Och and Ney, 2002).

\[
\Pr(x \mid y) = \frac{\exp \left\{ \sum_{m=1}^{M} \lambda_m h_m(x, y) \right\}}{\sum_{x'} \exp \left\{ \sum_{m=1}^{M} \lambda_m h_m(x', y) \right\}}
\]
Log-linear Models for Word Alignment

\[ P_r(a|e,f) = \frac{\exp \left\{ \sum_{m=1}^{M} \lambda_m h_m (a,e,f) \right\}}{\sum_{a'} \exp \left\{ \sum_{m=1}^{M} \lambda_m h_m (a',e,f) \right\}} \]

Log-linear models ARE statistical models.
Three Problems

• Feature selection
  – Which knowledge sources are useful and how to design feature functions to make use of them?

• Training
  – How to estimate the model scaling factors?

• Search
  – How to search the optimal alignment in an effective and efficient way?
Feature selection

• IBM translation model 3

\[ h(a, e, f) = Pr(f_i^l, a_i^l | e_i^l) \]

\[ = \left( \frac{m - \phi_0}{\phi_0} \right) p_0^{m-2\phi_0} p_1 \prod_{i=1}^{l} \phi_i! n(\phi_i | e_i) \times \]

\[ \prod_{j=1}^{m} t(f_j | e_{a_j}) d(j | a_j, l, m) \]

• POS tags transition model

\[ h(a, e, f, eT, fT) = \prod_a t(fT_a(j) | eT_a(i)) \]

• Bilingual dictionary coverage

\[ h(a, e, f, D) = \sum_a \text{occur}(e_a(i), f_a(j), D) \]
Training

- We use YASMET, which implement GIS algorithm, to train model scaling factors.
- We select the model parameters that yield best alignments on the development corpus.
- POS tags transition probabilities are also estimated on development corpus.

\[ p(fT|eT) = \frac{N_A(fT,eT)}{N(eT)} \]

Here, \( N_A(fT,eT) \) is the frequency that the POS tag \( fT \) is aligned to POS tag \( eT \) and \( N(eT) \) is the frequency of \( eT \) in the development corpus.
We use a greedy search algorithm to search the alignment with highest probability in the space of all possible alignments. A state in this space is a partial alignment. A transition is defined as the addition of a single link to the current state. A start state is the empty alignment. A terminal state is a state in which no more links can be added to increase the probability of current state.
An Example

我 是 一 个 学 生

I am a student
An Example

20 possible links!
An Example

I am a student

The partial alignment with the greatest probability

20 possible links!
An Example

我是一个学生
I am a student

我是一个学生
I am a student

19 possible links!
An Example

我是一个学生

I am a student

我是一个学生

I am a student

19 possible links!
An Example

I am a student

I am a student
An Example

Start state | Intermediate state | terminal state

我是一个学生  | 我是一个学生          | 我是一个学生

I am a student | I am a student        | I am a student

No links can be added to increase the probability of terminal state!
We compute $gain$, which is a heuristic function, instead of probability for efficiency.
Search Algorithm

Input: e, f, eT, fT, and D

Output: a

1. Start with a = φ.
2. Do for each l = (i, j) and l ∉ a:
   - Compute gain(a, l)
3. Terminate if ∀l, gain(a, l) ≤ 1.
4. Add the link ĝ with the maximal gain(a, l) to a.
5. Goto 2.
## Greedy Vs. Hill climbing

<table>
<thead>
<tr>
<th></th>
<th>Greedy</th>
<th>Hill climbing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operators</strong></td>
<td>Add (a special case of Move)</td>
<td>Move</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Swap</td>
</tr>
<tr>
<td><strong>Initial alignment</strong></td>
<td>empty alignment</td>
<td>Viterbi</td>
</tr>
<tr>
<td></td>
<td></td>
<td>alignment of a simple model</td>
</tr>
<tr>
<td><strong>Applicability</strong></td>
<td>log-linear models</td>
<td>fertility-based models</td>
</tr>
<tr>
<td><strong>Algorithm type</strong></td>
<td>greedy</td>
<td>greedy</td>
</tr>
</tbody>
</table>
Problems with the Search Algorithm

$$h(a, e, f) = Pr(f_1^J, a_1^J|e_1^I)$$

$$= \left( m - \phi_0 \right) p_0^{m-2\phi_0} p_1^{\phi_0} \prod_{i=1}^{l} \phi_i! n(\phi_i|e_i) \times$$

$$\prod_{j=1}^{m} t(f_j|e_{a_j}) d(j|a_j, l, m)$$

However, the search algorithm, which is general enough for any log-linear models, is not efficient for our models. It is time-consuming for each feature to figure out a probability when adding a new link, especially when the sentences are very long.
We restrict that $h_m(a,e,f) \geq 0$ for all feature functions. Note that we still call the new heuristic function $gain$ to reduce notational overhead. As a result, the termination condition will change to:

$$
\sum_{m=1}^{M} \lambda_m \log \left( \frac{h_m(a \cup l, e, f)}{h_m(a,e,f)} \right) \leq t
$$

$$
t = \sum_{m=1}^{M} \lambda_m \left\{ \log \left( \frac{h_m(a \cup l, e, f)}{h_m(a,e,f)} \right) - \left[ h_m(a \cup l, e, f) - h_m(a,e,f) \right] \right\}
$$

We call $t$ the gain threshold. It depends on the added link. But we remove this dependency for simplicity when using it in search algorithm by treating it as a fixed real-valued number.
Why we develop a new Gain?

• In the old gain, every feature has to figure out a probability; in the new gain, many terms will be cancelled out. For example, if a new link \( l=(i, j) \) is added, for IBM model 3 alone the new gain will only compute:

\[
\frac{p_0 \times p_0}{p_1} \times \frac{\phi_0 \times (m - \phi_0 + 1)}{(m - 2\phi_0 + 1) \times (m - 2\phi_0 + 2)} \times (\phi_i + 1) \times \\
\frac{n(\phi_i + 1 | e_i)}{n(\phi_i | e_i)} \times \frac{t(f_j | e_i)}{t(f_j | e_0)} \times d(j | i, l, m)
\]

The reason why we develop a new way to compute gain is that we try to reduce the computation.
New Search Algorithm

Input: $e, f, eT, fT, D$ and $t$

Output: $a$

1. Start with $a = \phi$.
2. Do for each $l = (i, j)$ and $l \notin a$:
   
   Compute $gain(a, l)$

3. Terminate if $\forall l, gain(a, l) \leq t$.
4. Add the link $\hat{l}$ with the maximal $gain(a, l)$ to $a$.
5. Goto 2.
How to get a n-best list?

• As shown above, we use a greedy search algorithm to search the optimal alignment. When we use GIS algorithm to train model scaling parameters, we need a n-best list. Therefore, we use a breadth-first search algorithm with pruning. During the search, every link will be added to every alignment in the n-best list and then update the n-best list.

• During the search, states those are indistinguish will be recombined
An illustration

N=2
- Start state
- Intermediate state
- Terminal state
- Discarded state
Experimental Results

Statistics of training corpus (Train), bilingual dictionary (Dict), development corpus (Dev) and test corpus (Test)

<table>
<thead>
<tr>
<th></th>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Train</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td></td>
<td>108 925</td>
</tr>
<tr>
<td>Words</td>
<td>3 784 106</td>
<td>3 862 637</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>49 962</td>
<td>55 698</td>
</tr>
<tr>
<td><strong>Dict</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entries</td>
<td></td>
<td>415 753</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>206 616</td>
<td>203 497</td>
</tr>
<tr>
<td><strong>Dev</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td></td>
<td>435</td>
</tr>
<tr>
<td>Words</td>
<td>11 462</td>
<td>14 252</td>
</tr>
<tr>
<td>Ave. SentLen</td>
<td>26.35</td>
<td>32.76</td>
</tr>
<tr>
<td><strong>Test</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td></td>
<td>500</td>
</tr>
<tr>
<td>Words</td>
<td>13 891</td>
<td>15 291</td>
</tr>
<tr>
<td>Ave. SentLen</td>
<td>27.78</td>
<td>30.58</td>
</tr>
</tbody>
</table>
Comparison of AER for results of using IBM Model 3 (GIZA++) and log-linear models

<table>
<thead>
<tr>
<th></th>
<th>Size of Training Corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1K</td>
</tr>
<tr>
<td>Model 3 E → C</td>
<td>0.4497</td>
</tr>
<tr>
<td>Model 3 C → E</td>
<td>0.4688</td>
</tr>
<tr>
<td>Intersection</td>
<td>0.4588</td>
</tr>
<tr>
<td>Union</td>
<td>0.4596</td>
</tr>
<tr>
<td>Refined Method</td>
<td>0.4154</td>
</tr>
<tr>
<td>Model 3 E → C</td>
<td>0.4490</td>
</tr>
<tr>
<td>+ Model 3 C → E</td>
<td>0.3970</td>
</tr>
<tr>
<td>+ POS E → C</td>
<td>0.3828</td>
</tr>
<tr>
<td>+ POS C → E</td>
<td>0.3795</td>
</tr>
<tr>
<td>+ Dict</td>
<td>0.3650</td>
</tr>
</tbody>
</table>
Comparison of AER for results of using IBM Model 5 (GIZA++) and log-linear models

<table>
<thead>
<tr>
<th></th>
<th>Size of Training Corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1K</td>
</tr>
<tr>
<td>Model 5 E → C</td>
<td>0.4384</td>
</tr>
<tr>
<td>Model 5 C → E</td>
<td>0.4564</td>
</tr>
<tr>
<td>Intersection</td>
<td>0.4432</td>
</tr>
<tr>
<td>Union</td>
<td>0.4499</td>
</tr>
<tr>
<td>Refined Method</td>
<td>0.4106</td>
</tr>
<tr>
<td>Model 3 E → C</td>
<td>0.4372</td>
</tr>
<tr>
<td>+ Model 3 C → E</td>
<td>0.3920</td>
</tr>
<tr>
<td>+ POS E → C</td>
<td>0.3807</td>
</tr>
<tr>
<td>+ POS C → E</td>
<td>0.3731</td>
</tr>
<tr>
<td>+ Dict</td>
<td>0.3612</td>
</tr>
</tbody>
</table>
Experimental Results (cont.)

Comparison on AER for various symmetrization methods: intersection, union, refined method and log-linear combination (i.e. M3 C->E + M3 E->C)
Experimental Results (cont.)

Effect of number of features and size of training corpus on search efficiency
### Experimental Results (cont.)

#### Resulting model scaling factors

<table>
<thead>
<tr>
<th>$\lambda_i$</th>
<th>MEC</th>
<th>+MCE</th>
<th>+PEC</th>
<th>+PCE</th>
<th>+Dict</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>1.000</td>
<td>0.466</td>
<td>0.291</td>
<td>0.202</td>
<td>0.151</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>-</td>
<td>0.534</td>
<td>0.312</td>
<td>0.212</td>
<td>0.167</td>
</tr>
<tr>
<td>$\lambda_3$</td>
<td>-</td>
<td>-</td>
<td>0.397</td>
<td>0.270</td>
<td>0.257</td>
</tr>
<tr>
<td>$\lambda_4$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.316</td>
<td>0.306</td>
</tr>
<tr>
<td>$\lambda_5$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.119</td>
</tr>
</tbody>
</table>
Experimental Results (cont.)

Precision, recall and AER over different gain thresholds with the same model scaling factors
Future Work

• Exploit more knowledge sources ranging from syntax-based models to various linguistic resources
• Optimize the model parameters directly with respect to AER
• Improve the efficiency of search algorithm
• Try on other language pairs
Thanks!