Direct Error Rate Minimization for Statistical Machine Translation

Tagyoung Chung
University of Rochester

Michel Galley
Microsoft Research
Summary of contributions

- Fast optimization of arbitrary feature weights is possible through direct minimization of error rate
Tuning in SMT
Statistical machine translation

- A log-linear model:

\[ P(e|f) \propto \exp \left( \sum_i \lambda_i h_i(e, f) \right) \]

feature weight  feature function

\( e: \) did not see the book you borrowed

\( f: \) 私は あなたが 借りた 本を 見なかった
Finding good weights

- Find $\lambda$ that minimizes error:

$$\hat{\lambda} = \arg\min_{\lambda} \mathbb{E} \left( r; \arg\max_e \exp \left( \sum_i \lambda_i h_i(e, f) \right) \right)$$

- Search within search makes problem hard
  - impossible to enumerate all target sentences
Common tuning framework

- Generate $N$-best list, add to the candidate pool, optimize $\lambda$, repeat the process with new $\lambda$
Problem with the framework

- \(N\)-best error surface is not the true error surface
- Most decoders make search errors
- Only linear model parameters \(\lambda\) are tuned
  - decoder parameters \(\theta\) are not tuned
  - cannot tune non-linear feature parameters
Direct search
Direct search

• Decoding and evaluation as a single function:

\[ \Phi(f, r; \lambda, \theta) = E\left(r; \arg\max_e \exp\left(\sum_i \lambda_i h_i(e, f)\right)\right) \]

• Use an optimization method to minimize the function:

\[ \arg\min_{\lambda, \theta} \Phi(f, r; \lambda, \theta) \]
Direct search

- Tunes all parameters
- Tunes the true error function (1-best BLEU)
- Used downhill simplex and Powell’s method
Downhill simplex

- Set up simplex on the search space
- Successively move simplex by replacing worse points with better ones
Powell’s method

- Use base vectors as a set of search directions
- Move towards the steepest direction (line searches)
- Discard the direction and add “average” direction
Parameters tuned with direct search
Distortion limit

- Number of words the decoder is allowed to skip
- too small: long-range reordering impossible
  too large: slower decoding, more search errors
- MERT cannot tune distortion limit
- Need to be tuned with log-linear model parameters
Polynomial features

- Total distortion penalty:
  \[
  D(e, f) = \lambda_d \sum_j \text{abs}(d_j)^{p_d}
  \]

- Linear penalty: \( p_d = 1 \), but \( p_d \) can be tuned

- Applicable to all features: \( h_i(e, f)^{p_i} \)
Future cost

\[ g(x) = \sum_i \lambda_i h_i(e, f) \]
\[ h(x) = \sum_i \lambda'_i h'_i(e, f) \]

- \( f(x) = g(x) + h(x) \)
  - \( f(x) \): estimated total cost
  - \( g(x) \): actual cost so far at current node \( x \)
  - \( h(x) \): estimated future cost

- Custom \( \lambda' \) enables better future cost estimation
\[ \lambda' = \lambda \text{ overestimates LM cost} \]
Search parameters

• Search-related parameters: beam size, histogram pruning, and threshold pruning

• Challenge: less pruning is usually preferred for better translation result
  – without time considerations, tuning might just learn to use larger beam size

• Solution: use time-sensitive evaluation metric
Search parameters

• Time-sensitive evaluation metric:
  – multiply time penalty to BLEU like brevity penalty
  – \( t_d \): desired decoding time
  – \( t_i \): current decoding time

\[
TP(\cdot) = \begin{cases} 
1.0 & t_i \leq t_d \\
\exp\left(1 - \frac{t_i}{t_d}\right) & t_i > t_d 
\end{cases}
\]

• Applicable to other resource constraints (e.g. memory)
Making direct search feasible
Challenges with direct search

- Function needs to be evaluated many times
  - function evaluation $\equiv$ re-decoding entire dev set

- Solution: speed-up search with following methods:
  - use significance test to quickly abandon unpromising point
  - use lattice-constrained decoding
• Models are evaluated incrementally in parallel

• Any model whose confidence interval does not overlap with the best one gets discarded
Statistical significance test

- Translate sentences in small batches and continue running approximate randomization test:
  - if point being tested becomes significantly worse, stop testing and move on
  - if not and difference between two points becomes small, stop testing and keep better one
Lattice-constrained decoding

- Store and reuse search graph
Lattice-constrained decoding

- Discards any translation hypotheses not in graph
- Decoder is not bypassed
  - decoder parameters can be tuned (e.g. smaller distortion limit than lattice is constructed with)
- Each edge stores all log-linear feature values
Experiments
Setup

- Decoder: Moses-like phrase-based decoder with standard set of features
- Data: Korean (7.9M), Arabic (11.1M), Farsi (739K)
- Dev/Test: 1K dev set, 6K test set (2K for Fas-Eng)
Setup

- Baseline: MERT with random restarts & distortion limit is tuned with grid search (from 3 to 10)
- Linear parameters are tuned in all experiments
- Uniform starting point for all experiments
- DL: distortion limit
  EX: future cost, polynomial features
  Search: beam, histogram pruning, threshold pruning
## Results

<table>
<thead>
<tr>
<th>Minimizer</th>
<th>parameters</th>
<th>Kor-Eng</th>
<th></th>
<th>Fas-Eng</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BLEU</td>
<td>Time</td>
<td>BLEU</td>
<td>Time</td>
</tr>
<tr>
<td>MERT</td>
<td>DL</td>
<td>23.3</td>
<td>20.8</td>
<td>32.2</td>
<td>11.7</td>
</tr>
<tr>
<td>simplex</td>
<td>DL</td>
<td>23.4</td>
<td>4.4</td>
<td>32.2</td>
<td>1.3</td>
</tr>
<tr>
<td>Powell</td>
<td>DL</td>
<td>23.4</td>
<td>5.6</td>
<td>32.3</td>
<td>2.1</td>
</tr>
<tr>
<td>Powell</td>
<td>EX, DL</td>
<td>23.6</td>
<td>8.9</td>
<td>32.5</td>
<td>4.9</td>
</tr>
<tr>
<td>Lattice-based</td>
<td>EX, DL</td>
<td>23.4</td>
<td>0.7</td>
<td>32.4</td>
<td>1.1</td>
</tr>
<tr>
<td>Powell</td>
<td>EX, DL, search</td>
<td>23.5</td>
<td>6.5</td>
<td><strong>32.6</strong></td>
<td>6.2</td>
</tr>
</tbody>
</table>
Contributions

• Direct minimization of the true error rate is feasible

• Tuning of decoder parameters is feasible

• Tuning non-linear features and hidden state features yields improvements
Thanks! Questions?