Improved Minimum Error Rate Training in Moses

Nicola Bertoldi, Barry Haddow and Jean-Baptiste Fouet

Third MT Marathon, Prague
28th January 2009
Outline

- MERT background
- The need for a new MERT in moses
- The design of the new MERT
- Evaluation
- Conclusions and future work
Discriminative Models

- State-of-the-art performance in statistical machine translation
- Combine outputs of several probabilistic models
- Features can be any function of source and target
  - e.g. forward/backward translation log probability, language model score, word penalty, etc.

Linear Model

\[ e^*(\lambda) = \arg \max_e \sum_{i=1}^{r} \lambda_i h_i(e, f) \]

feature weights \( \lambda_1 \ldots \lambda_r \) and feature functions \( h_1 \ldots h_r \).
Weight Optimisation

- How do we choose the best lambda?
  - We want the weights that produce the best translations.
  - Where “best” is measured by some automatic metric, eg BLEU, PER etc.
- Most popular method is minimum error rate training (MERT), proposed by Och (2003).
  - A form of coordinate ascent
  - Uses n-best lists from tuning set to approximate decoder output
  - Generally works well for small numbers of features (up to 20 or 30)
  - Implementation available in moses
The Need for a New MERT

- The existing moses MERT implementation has a number of issues
  - Lack of modularity in the design makes it difficult to e.g. replace BLEU with another automatic metric.
  - Mix of program languages in implementation hinders experimentation.

- At MTM2, a reimplementation of MERT was instigated with the following goals:
  - Clean, modular design to facilitate extension and experimentation
  - Separation of translation metric and optimisation code
  - Standalone open-source software, isolated from moses
  - Improved efficiency
MERT consists of inner and outer loop.

- Outer loop runs the decoder over the tuning set and produces n-best lists.
- Inner loop does weight optimisation.
- Iterate outer loop until convergence.
- Inner loop was replaced in the new MERT.
MERT Design: Inner loop

- Uses the n-best lists and references
  - N-best lists of previous iterations are merged
- Aims to find the weight set that maximises the translation score on the tuning set
- Consists of two main components:
  - Scorer Calculates translation metric
  - Optimiser Searches for the best weight set
- These are implemented as separate classes
- Can add a new Scorer/Optimiser by implementing new subclass
- For efficiency, some scoring statistics are pre-calculated in a separate extraction phase.
Evaluation: Translation performance on Heldout Test Sets

- Evaluation was performed on two different tasks from WMT08
- Standard moses system with 100-best lists for tuning
- Scores in tables are all $\text{BLEU}$

<table>
<thead>
<tr>
<th></th>
<th>nc-devtest07</th>
<th>nc-test07</th>
<th>newstest08</th>
</tr>
</thead>
<tbody>
<tr>
<td>old MERT</td>
<td>24.42</td>
<td>25.55</td>
<td>15.50</td>
</tr>
<tr>
<td>new MERT</td>
<td>24.87</td>
<td>25.70</td>
<td>15.54</td>
</tr>
</tbody>
</table>

Table: Comparison using the news commentary task.

<table>
<thead>
<tr>
<th></th>
<th>devtest06</th>
<th>test06</th>
<th>test07</th>
</tr>
</thead>
<tbody>
<tr>
<td>old MERT</td>
<td>32.75</td>
<td>32.67</td>
<td>33.23</td>
</tr>
<tr>
<td>new MERT</td>
<td>32.86</td>
<td>32.79</td>
<td>33.19</td>
</tr>
</tbody>
</table>

Table: Comparison using the europarl task.
This shows the variation of **BLEU** on tuning and heldout sets, against iterations of the outer loop.

Standard moses set up, with europarl training and WMT dev06 and test08 for tuning and heldout test, respectively.

Compare MERT old, with MERT new using 1,3 or all previous n-best lists.

- Note: ability to specify previous list count is new.

### Development

<table>
<thead>
<tr>
<th>BLEU (%)</th>
<th>iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>old</td>
<td>20</td>
</tr>
<tr>
<td>new-all</td>
<td>20.5</td>
</tr>
<tr>
<td>new-3</td>
<td>21</td>
</tr>
<tr>
<td>new-1</td>
<td>21.5</td>
</tr>
</tbody>
</table>

### Test

<table>
<thead>
<tr>
<th>BLEU (%)</th>
<th>iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>old</td>
<td>20</td>
</tr>
<tr>
<td>new-all</td>
<td>20.5</td>
</tr>
<tr>
<td>new-3</td>
<td>21</td>
</tr>
<tr>
<td>new-1</td>
<td>21.5</td>
</tr>
</tbody>
</table>
The graph below shows the on-disk usage of mert for each iteration.

- The new MERT implementation uses more disk as duplicate removal has not yet been implemented.
The graphs below compare execution time for old MERT and three different configurations of new MERT.

Total accumulated inner-loop time

Time for phase 1 (extraction) and phase 2 (optimisation).
New MERT concatenates latest n-best list to previous ones
Old MERT merges lists, removing duplicates
This means that new MERT has very short extraction phase, which does not increase with number of iterations
  But optimisation time increases more quickly with number of iterations
  It is roughly linear in the size of the combined n-best list
  And disk usage increases too

Duplicate removal should reduce execution time
Evaluation: Extensibility

- The principal aim of rewriting MERT was to provide a more flexible design
  - So it should be easier to incorporate new features
- Cer, Jurafsky and Manning (WMT 2008) showed how MERT could be improved by “regularisation”
  - Smoothing out of the error surface helps to avoid local maxima in the translation metric
  - This is done by either taking an average or minimum over a neighbourhood
- This smoothing was added to the scorer base-class
  - Making it available to any scorer
- The smoothing was tested on fr-en and de-en WMT08 europarl data.
### Smoothing Experiments: BLEU scores

<table>
<thead>
<tr>
<th>Method</th>
<th>Window</th>
<th>fr-en</th>
<th>de-en</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>devtest06</td>
<td>test06</td>
</tr>
<tr>
<td>none</td>
<td>n/a</td>
<td>32.86</td>
<td>32.79</td>
</tr>
<tr>
<td>minimum</td>
<td>±1</td>
<td>32.70</td>
<td>32.65</td>
</tr>
<tr>
<td></td>
<td>±2</td>
<td>32.81</td>
<td>32.75</td>
</tr>
<tr>
<td></td>
<td>±3</td>
<td>32.83</td>
<td>32.76</td>
</tr>
<tr>
<td></td>
<td>±4</td>
<td>32.88</td>
<td>32.77</td>
</tr>
<tr>
<td>average</td>
<td>±1</td>
<td>32.79</td>
<td>32.77</td>
</tr>
<tr>
<td></td>
<td>±2</td>
<td>32.89</td>
<td><strong>32.83</strong></td>
</tr>
<tr>
<td></td>
<td>±3</td>
<td>32.78</td>
<td>32.67</td>
</tr>
<tr>
<td></td>
<td>±4</td>
<td>32.81</td>
<td>32.79</td>
</tr>
</tbody>
</table>

- The gains of 0.5-1.0 BLEU reported by Cer et al. were not reproduced.
- They used a Chinese-English translation task.
- The error surface may have more noise.
Conclusions

- We have described a new open source implementation of MERT
- It is distributed within moses, but is standalone
- Modularity allows easy replacement of optimiser or translation metric
  - Currently both PER and BLEU scorers are available
- The translation performance of systems tuned by the new MERT is similar to those tuned by the old MERT
- The new MERT is slightly slower and uses more disk than the old
  - This is a known problem which will be rectified
Possible Enhancements

- Implement duplicate removal when merging n-best lists
- Add new automatic metrics eg WER, METEOR, combinations of metrics
- Add ability to constrain the feature weights
- Add priors to the weights
- Investigate parallelisation of the algorithm
- Implement the lattice optimisation proposed by Macherey et al (EMNLP 2008)
Thank you!
Questions?