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

$$\mathbf{e}^*(\lambda) = \arg\max_{\mathbf{e}} \sum_{i=1}^{r} \lambda_i h_i(\mathbf{e}, \mathbf{f})$$

feature weights $\lambda_i \ldots \lambda_r$ and feature functions $h_1 \ldots h_r$.

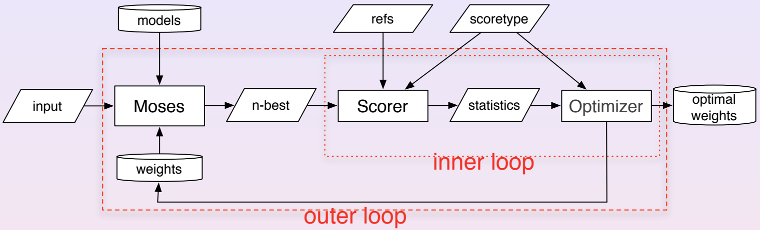
Weight Optimisation

- 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

Architecture of new MERT



- 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 BLEU

	nc-devtest07	nc-test07	newstest08
old MERT	24.42	25.55	15.50
new MERT	24.87	25.70	15.54

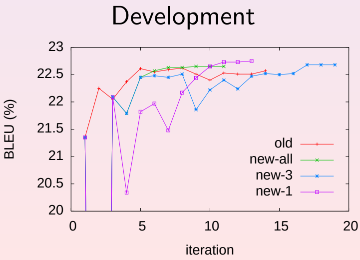
Table: Comparison using the news commentary task.

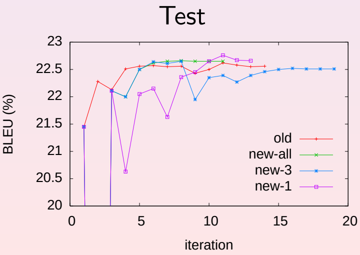
	devtest06	test06	test07
old MERT	32.75	32.67	33.23
new MERT	32.86	32.79	33.19

Table: Comparison using the europarl task.

Evaluation: Iterations

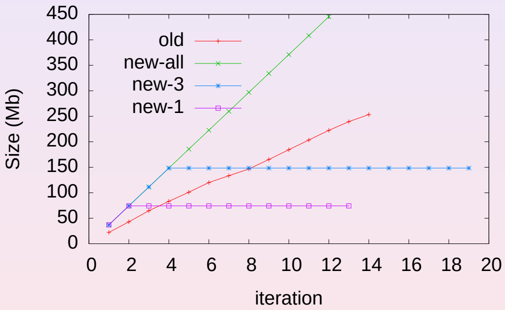
- This shows the variation of BLEU on tuning and heldout sets, against iterations of the outer loop
- Standard moses set up, with europarl training and WMT dev06 and test08 for tuning and heldout test, respectively.
- Compare MERT old, with MERT new using 1,3 or all previous n-best lists
  - Note: ability to specify previous list count is new





Evaluation: Disk usage

- The graph below shows the on-disk usage of mert for each iteration

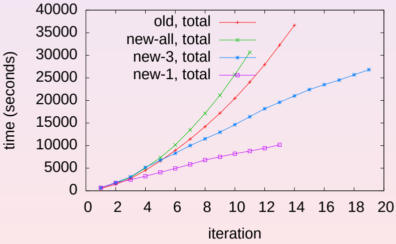


- The new MERT implementation uses more disk as duplicate removal has not yet been implemented

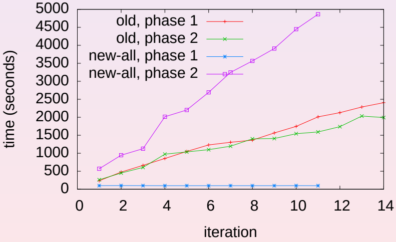
Evaluation: Execution time

- 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).



Evaluation: Execution Time - Comments

- 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

Method	Window	fr-en			de-en		
		devtest06	test06	test07	devtest06	test06	test07
none	n/a	32.86	32.79	33.19	27.54	27.67	28.07
minimum	±1	32.70	32.65	33.20	27.51	27.79	28.00
	±2	32.81	32.75	33.21	27.75	27.85	28.10
	±3	32.83	32.76	32.93	27.70	27.92	27.96
	±4	32.88	32.77	33.24	27.70	27.87	28.02
average	±1	32.79	32.77	33.29	27.44	27.81	28.00
	±2	32.89	32.83	33.28	27.63	27.73	27.98
	±3	32.78	32.67	33.19	27.53	27.67	27.87
	±4	32.81	32.79	33.25	27.81	28.01	28.22

- 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 automatric 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)

Questions?

Thank you!
Questions?