Margin Infused Relaxed Algorithm (MIRA) for Moses

Eva Hasler, Barry Haddow, Philipp Koehn

Institute for Language, Cognition and Computation, University of Edinburgh

September 7, 2011
1. Introduction
   - Background
   - Motivation

2. MIRA implementation for Moses
   - Selecting constraints
   - Main parameters
   - Stopping criterion and final weight selection
   - Parallelization
   - Usage

3. Experiments
   - MERT and MIRA results for models with core features
   - MIRA results for models with large feature sets
   - Parallelization
   - Start weights

4. Conclusions and Future work
Log-linear model

- typical core features of statistical machine translation (SMT) models: phrase translation model, language model, reordering model
- generative features as well as arbitrary features (no probabilistic interpretation), e.g. word or phrase penalty
- combined in a log-linear model → weighted score of all feature functions

\[
P(e,d|f) = \frac{\exp \sum_{k=1}^{K} \lambda_k h_k(e,d,f)}{\sum_{e',d'} \exp \sum_{k=1}^{K} \lambda_k h_k(e',d',f)}
\]
Adding features

- can improve discriminative power by adding more feature functions $h_k$
- more fine-grained, e.g. binary phrase features
- by assigning a weight $\lambda_i$ to each of them, let the parameter tuning algorithm choose useful features
- features growing in the thousands or millions pose a challenge for parameter tuning algorithms.

$$h_k(f_i, e_i) = \begin{cases} 
1, & \text{if } f_i=\text{“kleines Haus”} \text{ and } e_i=\text{“small house”} \\
0, & \text{otherwise}
\end{cases}$$
MIRA [Crammer and Singer, 2003]

- online large margin algorithm (originally for multi-class classification)
- ultra-conservative: weights are only updated when algorithm makes a mistake
- online update with margin-dependent learning rate
- margin can be tied to a loss function like BLEU
- tune model such that model score difference between two translations reflects the loss in BLEU between them
- important: selection of oracle translations and competing translations
Tuning weights with MIRA

Initialize: weight vector \( \mathbf{w} \)

Loop: For \( t = 1, 2, \ldots, T \) (\( T = \text{max. number of epochs} \))

- For all input sentences \( f_i \in \{f_1, \ldots, f_n\} \):
  - translate \( f_i \) with current weights \( \rightarrow \) n-best list(s) of \( e_i \)
  - select oracle translation \( e_i^* \) and competing translation(s) \( e_{ij} \)
  - form constraints of the form

\[
(h(e_i^*) - h(e_{ij})) \cdot \mathbf{w} \geq \text{loss}(e_i^*, e_{ij}) \quad \forall j
\]

- seek smallest update \( \mathbf{w}' \) subject to constraints

Output: averaged final weight vector \( \mathbf{w} \)
Constrained optimization problem

\[ w_{t+1} = \arg\min_w \frac{1}{2} \|w - w_t\|^2 + C \sum_j \xi_j \]

subject to

\[ \text{loss}_j - \Delta h_j \cdot w \leq \xi_j, \quad \forall j \in J \subseteq \{1, \ldots, m\} \]

Update rule

\[ w_{t+1} = w_t + \sum_j \alpha_j \Delta h_j \]

Solving for step size \( \alpha \) in case of a single constraint

\[ \alpha = \min \left\{ C, \frac{\text{loss} - \Delta h \cdot w}{\|\Delta h\|^2} \right\} \]
Motivation: Problems with Minimum Error Rate Training

- can only tune 15-30 parameters reliably
- needs reasonable start weights
- results vary considerably between different runs
Motivation: Problems with Minimum Error Rate Training

- can only tune 15-30 parameters reliably
- needs reasonable start weights
- results vary considerably between different runs

MIRA has been suggested for tuning MT system with larger feature sets

- [Arun and Koehn, 2007] explored training a phrase-based SMT system in a discriminative fashion with MIRA
- [Watanabe et al., 2007], [Chiang et al., 2009] added thousands of features to their baseline systems and tuned with MIRA
- need method for tuning feature-rich system within Moses toolkit for progress in feature engineering
MIRA implementation for Moses
Selecting constraints

Constraints for computing weight updates

• oracle and hypothesis selection (1): [Chiang et al., 2008]
  
  • 10-best list according to best model score
  • “good“ 10-best list (hope) according to
    \[ \hat{e} = \arg\max_e (\text{model score}(e) + \text{approx. BLEU score}(e)) \]
    (best from this list is oracle)
  • ”bad“ 10-best list (fear) according to
    \[ \hat{e} = \arg\max_e (\text{model score}(e) - \text{approx. BLEU score}(e)) \]
  • pair translations for all lists
Constraints for computing weight updates

- oracle and hypothesis selection (1): [Chiang et al., 2008]
  - 10-best list according to best model score
  - “good” 10-best list (hope) according to
    \[ \hat{e} = \arg \max_e (\text{model score}(e) + \text{approx. BLEU score}(e)) \]
    (best from this list is oracle)
  - ”bad“ 10-best list (fear) according to
    \[ \hat{e} = \arg \max_e (\text{model score}(e) - \text{approx. BLEU score}(e)) \]
  - pair translations for all lists

- oracle and hypothesis selection (2):
  use only the hope and fear lists
Constraints for computing weight updates

- oracle and hypothesis selection (1): [Chiang et al., 2008]
  - 10-best list according to best model score
  - "good" 10-best list (hope) according to
    \[ \hat{e} = \arg \max_e (\text{model score}(e) + \text{approx. BLEU score}(e)) \]
    (best from this list is oracle)
  - "bad" 10-best list (fear) according to
    \[ \hat{e} = \arg \max_e (\text{model score}(e) - \text{approx. BLEU score}(e)) \]
- pair translations for all lists

- oracle and hypothesis selection (2): use only the hope and fear lists

Solving optimization problems

- number and type of constraints can vary
- closed-form solution for update with single constraint
- Hildreth's algorithm for multiple constraints
Some parameters for MIRA training

--**hope-fear** (def: true), --**model-hope-fear** (def: false), 2 n-best lists or 3 n-best lists as mentioned above

--**nbest,n** size of n-best lists
Some parameters for MIRA training

--**hope-fear** (def: true), --**model-hope-fear** (def: false), 2 n-best lists or 3 n-best lists as mentioned above

--**nbest,n** size of n-best lists

--**shuffle** shuffle dev. set to avoid sequence bias (def: false)
Some parameters for MIRA training

--**hope-fear** (def: true), --**model-hope-fear** (def: false), 2 n-best lists or 3 n-best lists as mentioned above

--**nbest,n** size of n-best lists

--**shuffle** shuffle dev. set to avoid sequence bias (def: false)

--**average-weights** compute final weights over all seen weight vectors (def: false) or only those of the current epoch
Some parameters for MIRA training

--hope-fear (def: true), --model-hope-fear (def: false), 2 n-best lists or 3 n-best lists as mentioned above

--nbest,n size of n-best lists

--shuffle shuffle dev. set to avoid sequence bias (def: false)

--average-weights compute final weights over all seen weight vectors (def: false) or only those of the current epoch

--batch-size number of sentences processed as batch (def: 1)
Some parameters for MIRA training

--**hope-fear** (def: true), --**model-hope-fear** (def: false), 2 n-best lists or 3 n-best lists as mentioned above

--**nbest,n** size of n-best lists

--**shuffle** shuffle dev. set to avoid sequence bias (def: false)

--**average-weights** compute final weights over all seen weight vectors (def: false) or only those of the current epoch

--**batch-size** number of sentences processed as batch (def: 1)

--**slack** MIRA updates can be regularized (def: 0.01); smaller values mean more regularization, 0 means no regularization (parameter C in objective)
Some parameters for MIRA training

--**hope-fear** (def: true), --**model-hope-fear** (def: false), 2 n-best lists or 3 n-best lists as mentioned above

--**nbest,n** size of n-best lists

--**shuffle** shuffle dev. set to avoid sequence bias (def: false)

--**average-weights** compute final weights over all seen weight vectors (def: false) or only those of the current epoch

--**batch-size** number of sentences processed as batch (def: 1)

--**slack** MIRA updates can be regularized (def: 0.01); smaller values mean more regularization, 0 means no regularization (parameter C in objective)

--**sentence-bleu** (def: true), --**history-of-1best** (def: false) sentence-level BLEU (+1 for n>1) or approximate document-level BLEU using a history as suggested by [Chiang et al., 2008]
Stopping criterion and final weight selection

- MIRA stops when no update has been performed during a full epoch
- When during three consecutive epochs the sum of all updates in each dimension has not changed by more than a predefined value
- Possible to set a decreasing learning rate that reduces update size as training progresses
- **Final weights**: best weights according to performance on held-out set during 5-10 training epochs (further epochs do not seem to improve results)
Parallelization with iterative parameter mixing

- parallelization of online learning methods not straightforward, because updates build on top of each other sequentially

- *iterative parameter mixing*: [McDonald et al., 2010] proposed variation of parameter mixing strategy
Parallelization with iterative parameter mixing

- Parallelization of online learning methods not straightforward, because updates build on top of each other sequentially.
- **Iterative parameter mixing**: [McDonald et al., 2010] proposed variation of parameter mixing strategy.
- Training data is split into $n$ shards, $n$ processors.
- Each processor updates its weight vector only according to its shard.
Parallelization with iterative parameter mixing

- parallelization of online learning methods not straightforward, because updates build on top of each other sequentially
- \textit{iterative parameter mixing}: [McDonald et al., 2010] proposed variation of parameter mixing strategy
- training data is split into \( n \) shards, \( n \) processors
- each processor updates its weight vector only according to its shard
- resulting \( n \) weight vectors are mixed after each training epoch
- McDonald et al. showed that iterative parameter mixing yields performance as good as or better than training serially
Parallelization

- MPI used for parallelization (e.g. OpenMPI)
- mix parameters $n$ times per epoch
- 0: no mixing, average at the end
MIRA implementation currently located in sourceforge git repository git://mosesdecoder.git.sourceforge.net/gitroot/mosesdecoder/mosesdecoder, branch *miramerge*

To start MIRA, run:

```bash
mira -f moses.ini -i source-file -r reference-file or training-expt.perl -config expt.cfg -exec
```

- if jobs=n, $n > 1$ in config file, several mira processes are started with mpirun
- training script decodes heldout set with dumped weight file and computes BLEU score on heldout set
- caching of translation options should be switched off in moses.ini file ([use-persistent-cache] 0)
Data and experimental setup:

- news commentary corpus (≈85K/100K parallel sentences), nc-dev, nc-devtest, nc-test, news-test
- language pairs **en-de, en-fr, de-en**
- one oracle and one hypothesis translation per example (1 hope/1 fear)
- sentence-level BLEU (+1 for n-grams with \( n > 1 \))
- uniform start weights
- 8 parallel processors
MERT and MIRA results for models with 14 core features

<table>
<thead>
<tr>
<th>Lang. pair</th>
<th>BLEU(dev test)</th>
<th>σ</th>
<th>BLEU(test1)</th>
<th>BLEU(test2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>en-de</td>
<td>17.6</td>
<td>0.083</td>
<td>15.1</td>
<td>11.0</td>
</tr>
<tr>
<td>en-fr</td>
<td>28.2</td>
<td>0.045</td>
<td>15.2</td>
<td>17.7</td>
</tr>
<tr>
<td>de-en</td>
<td>26.5</td>
<td>0.082</td>
<td>22.9</td>
<td>15.5</td>
</tr>
</tbody>
</table>

Average results of 3 MERT runs

<table>
<thead>
<tr>
<th>Lang. pair</th>
<th>BLEU(dev test)</th>
<th>σ</th>
<th>BLEU(test1)</th>
<th>BLEU(test2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>en-de</td>
<td>17.7</td>
<td>0.013</td>
<td>14.9</td>
<td>11.1</td>
</tr>
<tr>
<td>en-fr</td>
<td>28.3</td>
<td>0.077</td>
<td>15.2</td>
<td>17.8</td>
</tr>
<tr>
<td>de-en</td>
<td>26.6</td>
<td>0.041</td>
<td>23.2</td>
<td>15.4</td>
</tr>
</tbody>
</table>

Average results of 3 shuffled MIRA runs (top: 10 epochs, bottom: 5)
Run times:

MERT using 8 threads:
10-21 hours for training (for 7-14 iterations)

MIRA using 8 parallel processors:
4 hours for 5 iterations, 8 hours for 10 iterations (plus some extra time for decoding devtest set)
MIRA results for models with large feature sets

<table>
<thead>
<tr>
<th>Lang. pair</th>
<th>en-de</th>
</tr>
</thead>
<tbody>
<tr>
<td>core features</td>
<td>17.7 (0.981)</td>
</tr>
<tr>
<td>core + word TB features</td>
<td>17.8 (0.984)</td>
</tr>
<tr>
<td>core + POS TB features</td>
<td>17.7 (0.986)</td>
</tr>
</tbody>
</table>

Average BLEU scores on dev. test set (3 MIRA runs) over 10 epochs, length ratio in brackets

- target word bigrams (TB): 33,300 active features
- POS bigrams: 1,400 active features
- comparable performance when training core + sparse features, possibly undertraining sparse features
### MIRA results for models with large feature sets

<table>
<thead>
<tr>
<th>Feature name</th>
<th>Feature weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distortion</td>
<td>0.207147</td>
</tr>
<tr>
<td>WordPenalty</td>
<td>-1.34204</td>
</tr>
<tr>
<td>LM</td>
<td>0.645341</td>
</tr>
<tr>
<td>dlmb_&lt;s&gt;:ART</td>
<td>0.247516</td>
</tr>
<tr>
<td>dlmb_&lt;s&gt;:NN</td>
<td>-0.10823</td>
</tr>
<tr>
<td>dlmb_ADJ:NN</td>
<td>0.137049</td>
</tr>
<tr>
<td>dlmb_NN:ADJ</td>
<td>-0.164686</td>
</tr>
</tbody>
</table>

Example feature weights of model with core + POS TB features

- `dlmb_<s>:ART` got positive weight, `dlmb_<s>:NN` got negative weight
  - model prefers German sentences starting with determiner
- model learned that adjective is likely to precede noun in German, not likely to follow noun
<table>
<thead>
<tr>
<th>Lang. pair</th>
<th># processors</th>
<th>Best BLEU(dev. test set)</th>
</tr>
</thead>
<tbody>
<tr>
<td>en-de</td>
<td>1</td>
<td>17.7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>17.7</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>17.7</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>17.7</td>
</tr>
<tr>
<td>en-fr</td>
<td>1</td>
<td>28.3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>28.4</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>28.2</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>28.3</td>
</tr>
<tr>
<td>de-en</td>
<td>1</td>
<td>26.6</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>26.6</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>26.6</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>26.5</td>
</tr>
</tbody>
</table>

- best results during 10 epochs, mixing frequency 5
- doubling number of processors reduces training time by half
- no systematic differences for varying number of processors
Start weights

<table>
<thead>
<tr>
<th>WP start</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>-0.3</td>
<td>-0.6</td>
<td>-0.9</td>
<td>-1.0</td>
<td>-1.1</td>
<td>-1.3</td>
<td>-1.3</td>
<td>-1.4</td>
<td>-1.5</td>
<td>-1.5</td>
</tr>
<tr>
<td>-1</td>
<td>-1.1</td>
<td>-1.2</td>
<td>-1.3</td>
<td>-1.4</td>
<td>-1.5</td>
<td>-1.5</td>
<td>-1.6</td>
<td>-1.6</td>
<td>-1.6</td>
<td>-1.7</td>
</tr>
</tbody>
</table>

Word penalty weight after each epoch, uniform vs. preset start weight

- MERT usually initialized with feature weights from past experience (lm=0.5, tm=0.2, wp=-1, ..)
- MIRA results were achieved with uniform start weights (0.1)
- weights become similar after some epochs
Start weights

<table>
<thead>
<tr>
<th>WP start</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>-0.3</td>
<td>-0.6</td>
<td>-0.9</td>
<td>-1.0</td>
<td>-1.1</td>
<td>-1.3</td>
<td>-1.3</td>
<td>-1.4</td>
<td>-1.5</td>
<td>-1.5</td>
</tr>
<tr>
<td>-1</td>
<td>-1.1</td>
<td>-1.2</td>
<td>-1.3</td>
<td>-1.4</td>
<td>-1.5</td>
<td>-1.5</td>
<td>-1.6</td>
<td>-1.6</td>
<td>-1.6</td>
<td>-1.7</td>
</tr>
</tbody>
</table>

Word penalty weight after each epoch, uniform vs. preset start weight

- MERT usually initialized with feature weights from past experience ($lm=0.5$, $tm=0.2$, $wp=-1$, ..)
- MIRA results were achieved with uniform start weights (0.1)
- weights become similar after some epochs
- best result with uniform start weights: BLEU=$17.68$
- best result with preset start weights: BLEU=$17.66$
- performance reached more quickly with preset start weights
Conclusions

- presented an open-source implementation of the Margin Infused Relaxed Algorithm for Moses toolkit
- reported results on core features sets and larger sparse feature sets
- showed that MIRA yields comparable performance to MERT with core features, can handle much larger feature sets
- can be run on parallel processors with negligible or no loss
- works well with uniform start weights
Conclusions

- presented an open-source implementation of the Margin Infused Relaxed Algorithm for Moses toolkit
- reported results on core features sets and larger sparse feature sets
- showed that MIRA yields comparable performance to MERT with core features, can handle much larger feature sets
- can be run on parallel processors with negligible or no loss
- works well with uniform start weights

Future work

- multi-threading
- validate for more language pairs and data sets
- more sparse features
Thank you!


