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

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

$$P(\mathbf{e},\mathbf{d}|\mathbf{f}) = \frac{\exp \sum_{k=1}^{K} \lambda_k h_k(\mathbf{e},\mathbf{d},\mathbf{f})}{\sum_{\mathbf{e'},\mathbf{d'}} \exp \sum_{k=1}^{K} \lambda_k h_k(\mathbf{e'},\mathbf{d'},\mathbf{f})}$$

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 λ_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..

$$h_k(f_i, e_i) = \begin{cases} 1, & \text{if } f_i = \text{``kleines Haus'' and } e_i = \text{``small house''} \\ 0, & \text{otherwise} \end{cases}$$

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 **w**

Loop: For $t = 1, 2, ..., T$ (T = max. number of epochs)

- For all input sentences $f_i \in \{f_1, .., f_n\}$:
- 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

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

- seek smallest update **w'** subject to constraints

Output: averaged final weight vector **w**

Constrained optimization problem

$$\mathbf{w}_{t+1} = \text{argmin}_{\mathbf{w}} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 + C \sum_j \xi_j$$

subject to

$$\mathit{loss}_j - \Delta \mathbf{h}_j \cdot \mathbf{w} \leq \xi_j, \quad \forall j \in J \subseteq \{1, .., m\}$$

Update rule

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \sum_j \alpha_j \Delta \mathbf{h}_j$$

Solving for step size α in case of a single constraint

$$\alpha = \min \left\{ C, \frac{\mathit{loss} - \Delta \mathbf{h} \cdot \mathbf{w}}{\|\Delta \mathbf{h}\|^2} \right\}$$

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

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
    $\hat{e} = \textbf{arg max}_{\textbf{e}}$ **(model score(e) + approx. BLEU score(e))**
    (best from this list is oracle)
  - ”bad“ 10-best list (*fear*) according to
    $\hat{e} = \textbf{arg max}_{\textbf{e}}$ **(model score(e) - 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
    $\hat{\mathbf{e}} = \mathbf{arg\ max_e\ (model\ score(e) + approx.\ BLEU\ score(e))}$
    (best from this list is oracle)
  - "bad" 10-best list (*fear*) according to
    $\hat{\mathbf{e}} = \mathbf{arg\ max_e\ (model\ score(e)\ \text{-}\ approx.\ BLEU\ score(e))}$
  - 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
    $\hat{\mathbf{e}} = \mathbf{arg\ max_e\ (model\ score(e) + approx.\ BLEU\ score(e))}$
    (best from this list is oracle)
  - "bad" 10-best list (*fear*) according to
    $\hat{\mathbf{e}} = \mathbf{arg\ max_e\ (model\ score(e) - approx.\ BLEU\ score(e))}$
  - 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

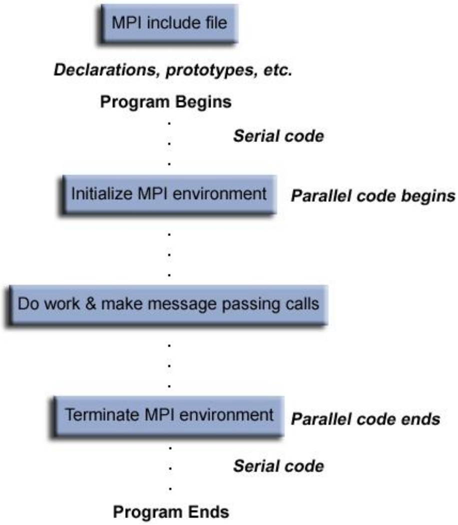
- parallelization of online learning methods not straightforward, because updates build on top of each other sequentially
- *iterative parameter mixing*: [McDonald et al., 2010] proposed variation of parameter mixing strategy

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
- *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



- MPI used for parellelization (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:
`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 ($\sim$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

Lang. pair	BLEU(dev test)	σ	BLEU(test1)	BLEU(test2)
en-de	17.6	0.083	**15.1**	11.0
en-fr	28.2	0.045	**15.2**	17.7
de-en	26.5	0.082	22.9	**15.5**

Average results of 3 **MERT** runs

Lang. pair	BLEU(dev test)	σ	BLEU(test1)	BLEU(test2)
en-de	**17.7**	0.013	14.9	**11.1**
en-fr	**28.3**	0.077	**15.2**	**17.8**
de-en	**26.6**	0.041	**23.2**	15.4
en-de	**17.6**	0.024	14.8	**11.2**
en-fr	28.0	0.059	**15.3**	**17.8**
de-en	**26.5**	0.039	**23.3**	15.3

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

Lang. pair	en-de
core features	17.7 (0.981)
core + word TB features	17.8 (0.984)
core + POS TB features	17.7 (0.986)

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

Feature name	Feature weight
Distortion	0.207147
WordPenalty	-1.34204
LM	0.645341
dlmb_<s>:ART	0.247516
dlmb_<s>:NN	-0.10823
dlmb_ADJ:NN	0.137049
dlmb_NN:ADJ	-0.164686

Example feature weights of model with core + POS TB features

- dlmb_<s>:ART got positive weight, dlmb_<s>:NN got negative weight
  $\rightarrow$ model prefers German sentences starting with determiner
- model learned that adjective is likely to preceed noun in German, not likely to follow noun

Lang. pair	# processors	Best BLEU(dev. test set)
en-de	1	17.7
	2	17.7
	4	17.7
	8	17.7
en-fr	1	28.3
	2	28.4
	4	28.2
	8	28.3
de-en	1	26.6
	2	26.6
	4	26.6
	8	26.5

- 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

WP start	1	2	3	4	5	6	7	8	9	10
0.1	-0.3	-0.6	-0.9	-1.0	-1.1	-1.3	-1.3	-1.4	-1.5	-1.5
-1	-1.1	-1.2	-1.3	-1.4	-1.5	-1.5	-1.6	-1.6	-1.6	-1.7

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

WP start	1	2	3	4	5	6	7	8	9	10
0.1	-0.3	-0.6	-0.9	-1.0	-1.1	-1.3	-1.3	-1.4	-1.5	-1.5
-1	-1.1	-1.2	-1.3	-1.4	-1.5	-1.5	-1.6	-1.6	-1.6	-1.7

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!

Arun, A. and Koehn, P. (2007).
Online Learning Methods For Discriminative Training of Phrase Based Statistical Machine Translation.
In *MT Summit XI, 2007*, Copenhagen.

Chiang, D., Knight, K., and Wang, W. (2009).
11,001 new features for statistical machine translation.
In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the NACL*, Stroudsburg, PA, USA. ACL.

Chiang, D., Marton, Y., and Resnik, P. (2008).
Online large-margin training of syntactic and structural translation features.
In *Proceedings of EMNLP 08*, Morristown, NJ, USA. ACL.

Crammer, K. and Singer, Y. (2003).
Ultraconservative online algorithms for multiclass problems.
Journal of Machine Learning Research, 3(4-5):951–991.

McDonald, R., Hall, K., and Mann, G. (2010).
Distributed Training Strategies for the Structured Perceptron.
In *Human Language Technologies: The 2010 Annual Conference of the NACL*, pages 456–464, Los Angeles, California. ACL.

Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. (2007).
Online large-margin training for statistical machine translation.
In *Proceedings of EMNLP-CoNLL*, pages 764–773, Prague. ACL.