PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 96 OCTOBER 2011 69–78

Margin Infused Relaxed Algorithm for Moses

Eva Hasler, Barry Haddow, Philipp Koehn

Institute for Language, Cognition and Computation, University of Edinburgh

Abstract

We describe an open-source implementation of the Margin Infused Relaxed Algorithm (MIRA) for statistical machine translation (SMT). The implementation is part of the Moses toolkit and can be used as an alternative to standard minimum error rate training (MERT). A description of the implementation and its usage on core feature sets as well as large, sparse feature sets is given and we report experimental results comparing the performance of MIRA with MERT in terms of translation quality and stability.

1. Introduction

1.1. Background

Statistical Machine Translation (SMT) systems usually consist of a number of models, each dealing with a particular aspect of the translation task. The core features of models like those described in (Koehn, 2010), i.e. phrase table, language model and reordering model, are likelihood-based features that are estimated in a generative fashion. These features are combined in a log-linear model as shown in equation (1), which produces a weighted score of all feature functions h_k given a source sentence $\mathbf{f}$, a target sentence $\mathbf{e}$ and a derivation $\mathbf{d}$.

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

Feature functions can consist of the generative features mentioned above but can also be arbitrary features whose values are not to be interpreted as probabilities, e.g. a word or phrase penalty. It is quite straightforward to improve the discriminative

Corresponding author: e.hasler@ed.ac.uk
Cite as: Eva Hasler, Barry Haddow, Philipp Koehn. Margin Infused Relaxed Algorithm for Moses. The Prague Bulletin of Mathematical Linguistics No. 96, 2011, pp. 69–78. doi: 10.2478/v10108-011-0012-3.

power of the model by adding feature functions h_k. For example we might want to use binary phrase features as in equation (2) to measure how much a particular phrase pair helps to discriminate between good and bad translations.

$$h_k(f_i, e_i) = \begin{cases} 1, & \text{if } f_i\text{="dieses Haus" and } e_i\text{="this house"} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

If we assign a weight λ_i to each of them, we can let the parameter tuning algorithm decide which features are useful for translation and which should be dropped. However, since the number of fine-grained features like these can easily grow in the thousands or millions, they pose a challenge for parameter tuning algorithms.

The Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003) is an online large margin algorithm that enforces a margin between different translations of the same sentence. This margin can be tied to a loss function which makes it straightforward to integrate BLEU (Papineni et al., 2002) or another quality measure. Given that we can provide the learning algorithm with good oracle translations, the model is tuned to score hypothesis translations with higher BLEU scores better than translations with lower BLEU scores. Picking oracle translations that represent good, reachable translations to update towards is an important part of the algorithm.

MIRA learns a weight $\mathbf{w}$ vector by additively updating the current decoder weights. After each new input sentence $f_i \in \{f_1, .., f_n\}$ was translated by the decoder, MIRA seeks the smallest update to the current weights subject to the following constraint. The difference in model scores, $\Delta\mathbf{h}_j \cdot \mathbf{w} = (\mathbf{h}(e_i^*) - \mathbf{h}(e_{ij})) \cdot \mathbf{w}$, between an oracle translation e_i^* and a hypothesis translation $e_{ij} \in \{e_{i1}, .., e_{im}\}$ must be at least as large as the loss $L(e_i^*, e_{ij}) = l_j$ between them. $\mathbf{h}(e_i)$ is a feature vector representation of translation e_i and the loss is defined as the difference in BLEU scores here but could be measured by other metrics as well.

The constrained optimization problem is illustrated in equation (3), where ξ is a non-negative slack variable[1], C is a positive *aggressiveness parameter* that controls the relative size of the update, t ranges over rounds of the algorithm and j ranges over a subset of hypothesis translations.

$$\begin{aligned} &\mathbf{w}_{t+1} = \text{argmin}_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C \sum_j \xi_j \\ &\text{subject to} \\ &l_j - \Delta\mathbf{h}_j \cdot \mathbf{w} \leq \xi_j, \quad \forall j \in J \subseteq \{1, .., m\} \end{aligned} \tag{3}$$

[1]Slack variables are introduced when the data are not linearly separable, see (Cortes and Vapnik, 1995).

Constructing the Lagrangian of equation (3) and taking partial derivatives yields the update rule in equation (4) which is defined in terms of the dual variables α_j. The vector $\boldsymbol{\alpha}$ of dual variables constitutes the step size for MIRA. Equation (5) shows how to solve for α if there is only one constraint in the optimization problem. The parameter C functions as an upper bound for α in the dual formulation[2]. For large optimization problems, $\boldsymbol{\alpha}$ can be found using iterative algorithms such as the Hildreth algorithm (Censor and Zenios, 1997) or SMO (Platt, 1998).

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \sum_j \alpha_j \Delta\mathbf{h}_j \tag{4}$$

$$\alpha = \min\left\{C, \frac{l - \Delta\mathbf{h} \cdot \mathbf{w}}{\|\Delta\mathbf{h}\|^2}\right\} \tag{5}$$

1.2. Motivation

Minimum Error Rate Training (MERT) is to date the most frequently used parameter tuning algorithm for SMT systems. It iteratively optimizes the parameters of an SMT model separately along each feature dimension and because of this, the number of parameters it can tune reliably ranges between only 15 to 30. The final weights depend on the given start weights which means that some prior knowledge about how a "good" set of feature weights looks like is advantageous. Another issue with MERT is that it uses random restarts and so the results can vary considerably between different runs.

MIRA has been suggested for tuning machine translation systems with larger features sets. Watanabe et al. (2007) and Chiang et al. (2009) added thousands of features to their baseline systems and showed improved translation quality after tuning the enhanced models with MIRA. Arun and Koehn (2007) explored training a phrase-based SMT system in a discriminative fashion with MIRA. McDonald et al. (2005) were the first to apply MIRA to train a dependency parser.

In order to promote further research for SMT systems in terms of feature engineering, it is important to have a method for tuning feature-rich models efficiently. The requirement for larger tuning sets when training sparse models suggests that online methods like MIRA will prove especially useful when scaling up discriminative training.

2. MIRA implementation for Moses

MIRA computes its weight updates by selecting hypothesis and oracle translations and solving an optimization problem with the constraints posed by these translations. Our implementation of MIRA solves a similar optimization problem to those of

[2] For details and derivations see (Crammer et al., 2006).

(Watanabe et al., 2007) and (Chiang et al., 2008) and was shown above in equation (3). The update for the optimization problem solved in every iteration of MIRA is computed with the Hildreth algorithm. The specific setup of the optimization problem can be controlled with parameters, for example, the number and type of hypothesis translations used for discriminative training can be varied.

Oracle translations are selected according to a modified decoder objective function: $\hat{e} = \arg\max_e$ (model score(e) + approx. BLEU score(e)). It is possible to use 3 different lists of hypothesis translations as suggested by Chiang et al., who used an n-best list according to the model score, a list of "good" translations (*hope*) according to the oracle selection objective and a list of "bad" translations (*fear*) according to $\hat{e} = \arg\max_e$ (model score(e) - approx. BLEU score(e)). Another option is to use only the *hope* and *fear* lists and leave out the n-best model translations.

The algorithm works by iterating over the training set sentence by sentence (or batch by batch), running the decoder on the current example to produce hypothesis translations. Given the resulting batch of translations, the feature vector representations are turned into constraints from which an update is computed and the algorithm moves on to the next example.

2.1. Main parameters

The following list shows the most important parameters for MIRA training. They were adapted from the literature about MIRA for SMT (mostly (Chiang et al., 2009), (Chiang et al., 2008), (Watanabe et al., 2007), (Arun and Koehn, 2007)). An epoch denotes a complete pass through the tuning data.

--hope-fear (def: true), **--model-hope-fear** (def: false), 2 n-best lists: constraints are formed between all pairs of *hope* and *fear* translations, or 3 n-best lists: each translation forms a constraint with the 1-best hope translation (oracle)

--learner Perceptron update or MIRA update (def: "mira")

--shuffle the development set may be shuffled to avoid sequence bias (def: false)

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

--batch-size number of input sentences processed as a 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-oracles**, **--history-of-1best** (def: false) sentence-level BLEU with smoothed precision counts for ngrams with n > 1 or approximate document-level BLEU using a history as suggested by Chiang et al.

--mixing-frequency see §2.3 for description (def: 5)

--perceptron-learning-rate (default: 0.01), **--mira-learning-rate** (def: 1) learning rates for weight vector updates

--scale-update scale MIRA update by oracle BLEU score (better oracles yield larger updates, def: false)

2.2. Stopping criterion and final weight selection

MIRA stops when no update has been performed during a full epoch and by default, it also stops when during three consecutive epochs the sum of all updates in each dimension has not changed by more than a predefined value (default: 0.01). This is supposed to capture the point when no more important updates to the weight vector are made and hence convergence was reached.

However, the selection of the final weights depends on MIRA's performance on a development test set that is translated using the current decoder weights during training (the number of times is configurable). According to our experiments, selecting the best weights found during 5-10 training epochs yields good results and the algorithm does not seem to improve further with more training epochs.

It is also possible to set a decreasing learning rate that reduces the size of the updates as the training progresses (parameter `--decr-learning-rate`).

2.3. Parallelization

MIRA can be run on multiple processors to speed up training times. In general, parallelization for online learning methods is not straightforward, because the updates build on top of each other sequentially. (McDonald et al., 2010) proposed a variation of a parameter mixing strategy that has proven useful for log-linear models, called *iterative parameter mixing*. It splits the training data into *shards* and each of n processors working in parallel updates its weight vector only according to the examples in its shard. After each training epoch, the resulting n weight vectors are mixed together using mixture coefficients. McDonald et al. showed for a named-entity recognition and a dependency parsing task that iterative parameter mixing yields performance as good as or better than training serially on all data.

Our implementation makes use of the Message Passing Interface (MPI). It follows the description of iterative parameter mixing and parameterizes the number of times per epoch the weight vectors are mixed across processors (no mixture coefficients are used). When the mixing frequency is set to 0, no mixing of the current weight vectors is performed but the accumulated weights from all past updates are still averaged across processors before dumping the average weights to disk.

2.4. Adding new feature functions

How to add new feature functions to Moses is described on the Moses website `http://www.statmt.org/moses/?n=Moses.FeatureFunctions`. When using score producers with an unlimited number of score components (features), it makes sense to keep their weights separate from the moses.ini file because in that case, we do not have a predefined set of feature IDs. Implementing those kinds of features is described in paragraph Moses.SparseFeatureFunctions.

2.5. Usage

The MIRA implementation is currently located at *mosesdecoder/branches/mira-mtm5/*. In order to use multiple processors for MIRA, it should be run from a training script[3] that starts the requested number of mira processes with mpirun. The script also picks up weight files dumped by MIRA to automatically translate a given development test set with the respective weights and compute its BLEU score. When using sparse features on top of the core features, the training script will separate them from the dumped weight files and put them into a separate weight file that can be passed to the Moses decoder with the parameter `-weight-file`.

Caching of translation options in Moses should be switched off in the moses.ini file used by MIRA (`[use-persistent-cache] 0`) in order to keep translations options always up to date with respect to the current decoder weights.

Available parameters can be printed with the `--help` flag. To start MIRA, run: `mira-mtm5/mira/mira -f moses.ini -i source-file -r reference-file` or `mira-mtm5/mira/training-expt.perl -config expt.cfg -exec` and pass parameters and the path to MIRA in the config file.

3. Experiments

All experiments where carried out on the news commentary corpus, with development set nc-dev2007, dev. test set nc-test2007 (for selecting final weights) and test sets nc-devtest2007 and news-test2008, see (Callison-Burch et al., 2008). Experiments were configured to use one oracle and one hypothesis translation (1 hope/1 fear) and sentence-level BLEU. Results are reported for language pairs en-de, en-fr and de-en. We show how the performance of MIRA compares to the performance of MERT on a core feature set and then show some results of models with additional large set of sparse features. We also report observations on parallelization and start weights.

3.1. MERT and MIRA results for models with core features

Tables 1 and 2 compare the performance of MERT and MIRA on a model with 14 core features. The results for MIRA were obtained by running the algorithm for 10 epochs on the original tuning set as well as well as two shuffled versions of it (3 runs) in order to test how much the final results depend on the order of the tuning set. The MERT results were obtained by averaging the results of 3 runs to account for the randomness within MERT. The tables show averaged values over the different runs as well as the standard deviation of BLEU and length ratio for the development test set. We can see that for 7 out of 9 compared BLEU scores, MIRA yields equivalent or better results, while the standard deviation shows that the order of the tuning sentences does not have a big impact on the final results for MIRA.

[3] *mira-mtm5/mira/training-expt.perl*, sample config file: *expt.cfg*

Lang. pair	Avg BLEU(dev test)	σ	Avg BLEU(test1)	Avg BLEU(test2)
en-de	17.6 (0.988)	0.083	**15.1** (0.966)	11.0 (1.046)
en-fr	28.2 (1.000)	**0.045**	**15.2** (1.125)	17.7 (1.016)
de-en	26.5 (1.012)	0.082	22.9 (1.048)	**15.5** (0.950)

Table 1. Average results of 3 MERT runs, news commentary data, length ratio in brackets

Lang. pair	Avg BLEU(dev test)	σ	Avg BLEU(test1)	Avg BLEU(test2)
en-de	**17.7** (0.981)	**0.013**	14.9 (0.957)	**11.1** (1.041)
en-fr	**28.3** (0.994)	0.077	**15.2** (1.119)	**17.8** (1.011)
de-en	**26.6** (1.000)	**0.041**	**23.2** (1.034)	15.4 (0.939)

Table 2. Average results of 3 MIRA runs (10 epochs), news commentary data, length ratio in brackets

Table 3 shows results on the two test sets when choosing the best weights from the first 5 epochs. The results are only slightly different (some even better) than the results for 10 epochs of training, which suggests that training for 5 epochs might already be sufficient. MERT using 8 threads on a machine with 8 CPUs took 10-21 hours for training (for 7-14 iterations), MIRA using 8 parallel processors took 4 hours for 5 iterations and 8 hours for 10 iterations. If a development test set is used to determine the final weight vector, some extra resources are needed for decoding that set.

3.2. MIRA results for models with large feature sets

Table 4 compares results on the development test set for a model with core features and two models with sparse target bigram (TB) features, one using word bigrams with 33,300 active features and the other using POS bigrams with 1,400 active features. Even though these features do not seem to improve translation performance very much[4], the results show that MIRA can deal with a large number of features and manages to train the core weights properly at the same time.

[4] Also the results on the test sets varied only slightly (less than $\pm$ 0.1 BLEU).

Lang. pair	Avg BLEU(dev test)	σ	Avg BLEU(test1)	Avg BLEU(test2)
en-de	**17.6** (0.970)	**0.024**	14.8 (0.945)	**11.2** (1.031)
en-fr	28.0 (0.987)	0.059	**15.3** (1.112)	**17.8** (1.005)
de-en	**26.5** (0.997)	**0.039**	**23.3** (1.030)	15.3 (0.936)

Table 3. Average results of 3 MIRA runs (5 epochs), news commentary data, length ratio in brackets

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)

Table 4. Average BLEU scores on dev. test set (avg. of 3 MIRA runs) over 10 epochs, news commentary data, length ratio in brackets.

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

Table 5. Example feature weights of model with core + TB features

Table 5 shows some of the core feature weights and the weights of some frequently occurring sparse features. We can see that the bigram feature dlmb_<s>:ART got a positive weight while the feature dlmb_<s>:NN got a negative weights, which means that the model prefers German sentences starting with articles to those starting with nouns. The model also learned that an adjective is likely to preceed a noun in German while it is not likely to follow a noun.

3.3. Parallelization

We ran experiments with different numbers of processors to investigate the variance between different parallel setups. Table 6 shows results for the same MIRA setup with 1, 2, 4, and 8 processors, all with a mixing frequency of 5. Since parallelization works by dividing the development set into shards that are sent to different processors, doubling the number of processors reduces the training time by half. Even though there is some variation in the results, we cannot observe a general tendency across language pairs that increasing the number of processors would change the results in a systematic way. The difference in BLEU scores within the same language pair was at most 0.2 which can be considered a negligible loss.

3.4. Start weights

MERT is usually initialized with feature weights that yield better performance than uniform weights according to past experience. The reported results for the MIRA experiments were achieved with uniform start weights (all weights 0.1), while the MERT experiments were initialized with the following weights: language model=0.5, distortion/reordering=0.3, translation features=0.2, word penalty=-1.

In the experiment reported in table 7, the development of the word penalty weight for uniform and preset start weights is shown. The reported values were measured at the end of each epoch, for 10 epochs. Even though the uniform weight started at

Lang. pair	# processors	Best BLEU(dev. test set)
en-de	1	17.7 (0.985)
	2	17.7 (0.977)
	4	17.7 (0.975)
	8	17.7 (0.973)
en-fr	1	28.3 (1.002)
	2	28.4 (0.997)
	4	28.2 (1.002)
	8	28.3 (0.999)
de-en	1	26.6 (1.007)
	2	26.6 (1.007)
	4	26.6 (1.005)
	8	26.5 (1.003)

Table 6. Varying the number of processors with a mixing frequency of 5, best results on dev. test set during 10 epochs, length ratio in brackets

WP start	1	2	3	4	5	6	7	8	9	10
0.1	-0.26	-0.60	-0.85	-1.00	-1.14	-1.25	-1.33	-1.41	-1.50	-1.54
-1	-1.09	-1.22	-1.32	-1.41	-1.46	-1.53	-1.58	-1.61	-1.64	-1.67

Table 7. Word penalty weight after each of 10 epochs, for uniform and preset start weights.

0.1 and the preset weight at -1, they became quite similar after a few epochs. The best result on the development test set was BLEU=17.68 with uniform start weights and BLEU=17.66 with preset start weights which shows that uniform initialization does not harm the performance of MIRA. However, after the first epoch, the development test performance was BLEU=17.14 for uniform start weights and BLEU=17.65 for preset start weights, indicating that good start weights result in shorter training times.

4. Conclusions

We presented an open-source implementation of the Margin Infused Relaxed Algorithm for the Moses toolkit. We reported results on core feature sets as well as large, sparse feature sets, showing that MIRA yields comparable performance to MERT on the core features and is able to handle much larger feature sets. We have also given evidence that MIRA can be run on parallel processors with negligible or no loss and that it works well with uniform start weights. In the future we want to scale up to larger training sets and explore new ways of integrating sparse features.

Bibliography

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

Callison-Burch, C., C. Fordyce, P. Koehn, C. Monz, and J. Schroeder. Further meta-evaluation of machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 70–106, Columbus, Ohio, June 2008. ACL.

Censor, Y. and S. A. Zenios. *Parallel Optimization - Theory, Algorithms, and Applications*. Oxford University Press, New York/Oxford, 1997.

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

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

Cortes, C. and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

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

Crammer, K., O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:551–585, 2006.

Koehn, P. *Statistical Machine Translation*. Cambridge University Press, 2010.

McDonald, R., K. Crammer, and F. Pereira. Online Large-Margin training of dependency parsers. In *Proceedings of ACL*, 2005.

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

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL 02: Proceedings of the 40th Annual Meeting of the ACL*, pages 311–318, Morristown, NJ, USA, July 2002. ACL.

Platt, J. C. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, 1998.

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

Address for correspondence:
Eva Hasler
e.hasler@ed.ac.uk
Informatics Forum, 10 Crichton Street
Edinburgh EH8 9AB, UK