IJCLA VOL. 4, NO. 2, JUL-DEC 2013, PP. 45–62
RECEIVED 07/12/12 ACCEPTED 04/03/13 FINAL 06/03/13

Fast Large-Margin Learning for Statistical Machine Translation

GUILLAUME WISNIEWSKI AND FRANÇOIS YVON

Univ. Paris Sud, France

ABSTRACT

Statistical Machine Translation (SMT) can be viewed as a generate-and-select process, where the selection of the best translation is based on multiple numerical features assessing the quality of a translation hypothesis. Training *a SMT system consists in finding the right balance between these features, so as to produce the best possible output, and is usually achieved through Minimum Error Rate Training (MERT). Despite several improvements, training remains one of the most time consuming step in the development of SMT systems and is a major bottleneck for experimentations. Building on recent advances in stochastic optimization and online machine learning, this paper studies a possible alternative to MERT, based on standard and well-understood algorithms. This approach is shown to deliver competitive solutions, at a much faster pace than the standard training machinery.*

1 INTRODUCTION

A statistical machine translation (SMT) system consists of a ruleset and a scoring function. The ruleset, represented either in the phrase table of a phrase-based system or in the rewrite rules of a hierarchical system, generates a set of translation hypotheses for each source sentence. These candidates are then ranked according to a scoring function so designed that the top ranking translation is also the best according to some external quality measure.

In the vast majority of existing SMT systems, the score of a hypothesis is computed as a linear combination of various numerical features.

The vector of coefficients, one for each feature, is learned using a training set made of source sentences and their accompanying translation reference(s), by maximizing some empirical gain over the training set, where the gain, for instance the BLEU score, evaluates the quality of the translation hypotheses obtained for a given weight vector.

Training of a SMT system is made difficult by the form of the inference rule used to compute hypotheses, the typical gains used in MT evaluation that are neither convex nor differentiable and the size of the search space that makes direct optimization intractable. Various heuristic optimization strategies have therefore been put forward, the most successful to date being MERT [1]. In this approach, optimal weights are derived through a complex iterative procedure which repeatedly: *i)* given the weights, decodes the training set to compute an approximation of the search space and *ii)* given this approximated search space, computes an optimal value for the weights.

If MERT has proven to be a practical and effective training procedure, it has been criticized on various grounds, notably for its inability to find good and stable solutions, especially when the feature vector exceeds a dozen dimensions. The computational cost of MERT, due to the need to repeatedly translate the training set, is also viewed as a serious issue: typical runs of MERT can take hours, sometimes days to complete.

Replacing MERT therefore remains a matter of active research. For instance, [2] reports experiments with several variants of MERT, aimed at making its results more stable. Another line of research has been to improve the approximation of the search space, using lists of randomly generated hypotheses [3], word lattices or derivation forests [4]. Inspired by recent advances in structured learning [5], the proposals of [6] and [7] are more radical and replace the gain with training criteria that are easier to optimize. Finally, the recent work of [8] recasts training as a learning-to-rank problem. The main motivation of all these studies was to increase the number of features used during learning, speed being a less important goal.

By contrast, the approach advocated in this work primarily aims at reducing the total training time, which is currently a significant bottleneck for experimentations. Like in [6], an important component of this proposal is the use of a large-margin learning criterion. We depart from existing large margin approaches to SMT by the use of lattices, from which promising pseudo-references (oracles) are efficiently extracted, and the recourse to fast stochastic optimization techniques. The main contribution of this work is to demonstrate, by putting all these ingredients to-

gether, that a large scale SMT system can be trained in only a few minutes, the number of decoding passes over the training set being reduced by a factor of almost ten. As discussed below, other advantages of our implementation are its simplicity, especially when compared to [7], and its theoretical guarantees which derive from convex optimization results. As a consequence, our approach does not suffer from stability issues, even for large feature sets.

The rest of the paper is organized as follows. We introduce the large-margin criterion in Section 2 and show how the resulting optimization problem can be easily solved using a subgradient method in Section 3. The optimization procedure is detailed in Section 4. Section 5 presents several MT experiments that show how fast our method is. Related works are summarized in Section 6 and we conclude in Section 7.

2 Large Margin Learning for SMT

2.1 *Notations*

The basic resource for training a SMT system is a training set $D = \{(s_i, r_i)\}_{1 \leq i \leq N}$, made of N source sentences s_i, each accompanied with a reference translation r_i. The set of possible translations for a sentence s_i will be denoted $\mathcal{H}_{s_i} = (\mathbf{h}_{i,j})_{1 \leq j \leq n_i}$. The search space of the decoder is often approximated by an explicit list of n-best hypotheses or by a lattice, which encodes compactly a larger number of potential translations.

Abusing notations, we will denote by $\mathbf{h}_{i,j}$ both a hypothesis (a sequence of words) and its feature representation. Given the search space $\mathcal{H}_{s_i}$ and a weight vector $\mathbf{w}$, translating a sentence s_i thus amounts to solving:

$$\mathbf{h}_i^* = f(s_i; \mathbf{w}) = \arg\max_{\mathbf{h} \in \mathcal{H}_{s_i}} \langle \mathbf{h} | \mathbf{w} \rangle \tag{1}$$

where $\mathbf{h}_i^*$ is the predicted translation and $\langle \cdot | \cdot \rangle$ is the dot product in $\mathbb{R}^d$. Using these notations, training a SMT system is the task of finding a weight vector $\mathbf{w}$ such that the predicted translations are as good as possible. Formally, training thus aims to solve the following problem:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} G(D; \mathbf{H}) \tag{2}$$

where the gain function G, for instance the BLEU score, evaluates the quality of the hypotheses $\mathbf{H} = \{\mathbf{h}_i^*, s_i \in D\}$ obtained for a given $\mathbf{w}$.

2.2 *Learning Criterion*

Regularized empirical risk minimization is a popular learning criterion that has proven effective in many applications. Applying it to learn the scoring function of a SMT system amounts to solving:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_{i=1}^{N} \ell^{\text{smt}}\left(f(s_i;\mathbf{w}), r_i\right), \tag{3}$$

where $\ell^{\text{smt}}(\mathbf{h}, r)$ is any *sentence level* loss that evaluates the quality of a hypothesis $\mathbf{h}$ with respect to a reference r, (s_i, r_i) is the i-th example, $f(s_i;\mathbf{w})$ is the prediction of the system. The first term of the objective is a regularizer that prevents overfitting and the second is the empirical risk (error on the train set). The hyper-parameter λ controls the strength of the regularization.

Direct optimization of (3) is generally not possible as usual SMT metrics are piecewise constant and therefore not differentiable. However, *structure learning* offers several ways to reformulate this problem in terms of convex programming by deriving upper bounds of arbitrary loss functions thanks to techniques such as margin-rescaling [9] or slack-rescaling [10]. While these upper bounds are not *consistent*, they have achieved optimal prediction accuracy in several tasks. In the following, we will describe the margin-rescaling technique as it can be implemented more easily than slack-rescaling. As detailed in Section 6, the resulting learning criterion is similar to the one optimized by MIRA.

2.3 *Margin Rescaling*

Consider the following generalization of the Hinge loss for the i-th example [9]:

$$\ell_i(\mathbf{w}) = \max_{2\leq j} \left(\ell^{\text{smt}}\left(\mathbf{h}_{i,j}, \mathbf{h}_{i,1}\right) - \langle \mathbf{w} | \mathbf{h}_{i,1} - \mathbf{h}_{i,j}\rangle\right) \tag{4}$$

This loss is convex (as a maximum over a family of linear functions) but is not differentiable everywhere; it is also obviously an upper-bound of $\ell^{\text{smt}}\left(\mathbf{h}_{i,j}, \mathbf{h}_{i,1}\right)$. It results from the following reformulation of the general large-margin classification problem: learning aims at finding a function that scores the correct output $\mathbf{h}_{i,1}$ higher than all other possible outputs $\mathbf{h}_{i,j}$ by a given margin. The worse the prediction of $\mathbf{h}_{i,j}$ compared

to $\mathbf{h}_{i,1}$, the larger the margin has to be, which is reflected by scaling the margin by $\ell^{\text{smt}}(\mathbf{h}_{i,1}, \mathbf{h}_{i,j})$ as follows:

$$\langle \mathbf{h}_{i,1} | \mathbf{w} \rangle + \xi_i \geq \langle \mathbf{h}_{i,j} | \mathbf{w} \rangle + \ell^{\text{smt}}(\mathbf{h}_{i,j}, \mathbf{h}_{i,1}) \quad \forall j \geq 2$$

where ξ_i is a slack variable. There are as many constraints as there are possible translations of the source. It is however possible to combine all these linear constraints in a single non-linear constraint:

$$\langle \mathbf{h}_{i,1} | \mathbf{w} \rangle + \xi_i \geq \max_{j \geq 2} \left(\langle \mathbf{h}_{i,j} | \mathbf{w} \rangle + \ell^{\text{smt}} \left(\mathbf{h}_{i,j}, \mathbf{h}_{i,1} \right) \right)$$

Moving the constraints of all examples to the objective of the large margin problem as described in [10] is a simple way to create a convex objective in $\mathbf{w}$ and recover the loss introduced in Equation (4). It must be stressed that, while margin-rescaling (as well as slack-rescaling) offers a generic way to derive a convex upper bound of an arbitrary loss function ℓ, the quality of this bound (how close it is to the "original" loss function) highly depends on the task and the loss function considered.

3 Optimization Procedure

Using the convex upper bound of the evaluation criterion ℓ^{smt} derived in the previous section, large-margin learning for SMT amounts to optimizing:

$$\min_{\mathbf{w}} \frac{\lambda}{2} ||\mathbf{w}||^2 + \frac{1}{n} \sum_{i=1}^{n} \ell_i(\mathbf{w}) \tag{5}$$

where $\ell_i(\mathbf{w})$ is defined in Equation (4).

Several methods have been proposed to solve this optimization problem [9, 10]. Following [11] we propose to solve it using a straight-forward subgradient descent method which can be easily implemented. Subgradient is a generalization of gradient to convex functions that are non-differentiable [12] and can be used in the same way as a gradient to optimize a function.

3.1 *Subgradient Optimization*

One subgradient of the objective (5) is given by:

$$\mathbf{g} = \lambda \cdot \mathbf{w} + \frac{1}{n} \sum_{i=1}^{n} \mathbf{h}_{i,j^*} - \mathbf{h}_{i,1} \tag{6}$$

where:

$$\mathbf{h}_{i,j^*} = \arg\max_j \langle \mathbf{h}_{i,j} | \mathbf{w} \rangle + \ell^{\text{smt}}(\mathbf{h}_{i,j}, \mathbf{h}_{i,1}) \quad (7)$$

The expression of $\mathbf{g}$ results from the following properties of a subgradient: *i)* a subgradient is linear; *ii)* if f is differentiable, its only subgradient is the gradient vector itself; *iii)* a subgradient of $\max_y f(x,y)$ is $\nabla_x f(x,y^*)$ for any $y^* \in \arg\max_y f(x,y)$ if f is differentiable with respect to x.

Computing the subgradient related to the i-th example requires solving the so-called *loss-augmented* problem described by Equation (7) and to find the best (oracle) hypothesis $\mathbf{h}_{i,1}$ according the evaluation metric ℓ^{smt}. These two problems are well-defined and, as described in Section 4.2, they can be solved efficiently. As a consequence, implementing this training strategy does not depend on any heuristic design decision, contrary to most existing large margin approaches to SMT.

Subgradient descent can be applied either in a *batch* setting in which parameter updates are performed on the basis of the (sub)gradient information accumulated over the entire training set or in a *online* or *stochastic* setting, in which parameters are updated on the basis of a single example chosen randomly at each iteration. In this case, the expression of $\mathbf{g}$ is simplified as the sum in Equation (6) vanishes.

Even though batch subgradient descent is known to be a slow optimization technique, using it in an online setting leads to fast convergence [13]. That is why, we only considered the online method. However, for stochastic descent, usual methods to find the optimal value of the learning rate, like line search, can not be applied and the learning rate sequence has to be chosen in advance. The optimization procedure is summarized in Algorithm 1.

3.2 *Averaged Stochastic Descent*

While online algorithms can converge to the neighborhood of the optimum very quickly, there are no guarantees that the objective function decreases after each update. Indeed updates are based only on a (noisy) estimate of the true gradient evaluated from a single example and might sometimes point to a wrong direction. This problem is of more importance in subgradient descent as a subgradient is not always a descent direction. That is why, in the learning curves representing the evolution of the objective function with respect to the number of iterations, the value

Algorithm 1: Optimization procedure

input : a number of iterations T and a sequence of learning rate η_t
w= NullVector()
for $t \in [\![1, T]\!]$ **do**
  pick an example (s, r) randomly
  compute $\mathbf{h}_{i,1} = \arg\max_{\mathbf{h} \in \mathcal{H}_s} \ell^{\text{smt}}(\mathbf{h}, r)$
  compute $\mathbf{h}_{i,j^*}$ according to Equation (7)
  update = $\lambda \cdot w + \mathbf{h}_{i,j^*} - \mathbf{h}_{i,1}$
  w = w - $\eta_t \times$ update
end

of the objective function is often observed to wobble around the optimum [14].

One practical way to reduce the fluctuations of the objective function is to *average* the weights over time. Several recent works [15, 16] have shown that *averaged* stochastic gradient descent leads to very fast convergence when the learning rate is set according to their guidelines: in some of their experiments, the optimum is reached after only a single pass over the train set even for large-scale problems.

4 Implementing Subgradient Descent

Implementing the optimization procedure described in the previous section requires us to define a suitable loss function ℓ^{smt} and to efficiently solve both the loss-augmented and the oracle decoding problems. These choices are described below.

4.1 *Loss Function*

Large-margin learning for SMT relies on a loss function ℓ^{smt} to evaluate the quality of a hypothesis with respect to a given reference *at the sentence-level*. Most of the metrics usually used for MT evaluation, such as BLEU or METEOR are computed at the corpus level. Moreover, contrary to these metrics, learning theory assumes that the smaller the loss is, the better the solution, the loss being 0 when the correct answer is predicted.

Several sentence-level approximation of the wide-spread BLEU metric have already been proposed [7], but we used a simpler approximation

that enforces the properties of a loss. Our approximation is based on a linear combination of the i-gram precision:

$$\text{score}(\mathbf{h}, r) = \sum_{i=1}^{I} \Xi_i \cdot c_i(\mathbf{h}, r) - \Xi_0 \cdot c_{\text{non}}(\mathbf{h}, r) \tag{8}$$

where $c_i(\mathbf{h}, r)$ is the number of common i-gram in the hypothesis $\mathbf{h}$ and in the reference r, c_{non} is the number of words of the hypothesis that do not appear in the reference and the Ξ_i are positive constants chosen to maximize the correlation between the BLEU score and its approximation.

The score defined by Equation (8) is a compromise between the number of words that the hypothesis and the reference have in common (accounting for the recall) and the number of words of the hypothesis that do not appear in the reference (accounting for the precision). It can be transformed into a loss: $\ell^{\text{smt}}(\mathbf{h}, r) = \alpha - \text{score}(\mathbf{h}, r)$ where α is the score of the best hypothesis. Computing α is needed since our approximation of BLEU is not normalized.

4.2 *Solving the Oracle Decoding and Loss-Augmented Problems*

For a given source sentence, the search space of a SMT system has the form of a directed acyclic graph (a lattice) in which each edge is associated with a phrase and a vector of features describing the cost of emitting this phrase. For simplicity, we assume that there is a single initial state and a single final state. Each path from the initial to the final state in this lattice corresponds to a translation hypothesis ; its feature representation can be worked out by summing the features on the edges and its "output string" by concatenating the phrases of the edges.

Many SMT problems, including the one appearing in Algorithm 1, can be formulated as shortest path problems in a lattice. For instance, the decoding task, described in Equation (1), is the shortest path problem in which the cost of an edge is defined by the opposite of the dot product between the feature representation of edge and the weight vector $\mathbf{w}$. As lattices are acyclic graphs, shortest path problems can be efficiently solved in a time linear in the number of edges and vertices.

Oracle decoding, the task of finding the best hypothesis according to the loss function, can also be performed using a shortest path algorithm, as long as the evaluation metric factorizes in terms of individual edges

[17]. Considering the BLEU-1 approximation introduced in Section 4.1, finding $\mathbf{h}_{i,1}$ amounts to solving:

$$\arg\min_{\boldsymbol{\pi} \in \Pi} - \sum_{i=1}^{n} \theta_{\pi_i}$$

where $\boldsymbol{\pi}$ is a path made of m edges $(\pi_i)_{i=1}^{m}$ in the lattice, Π is the set of all paths and θ_{π_i} is the cost of the edge π_i. It is defined by $\theta_{\pi_i} = \Xi_1 \times c_1(w, r) - \Xi_0 \times c_{\text{non}}(w, r)$ where w is the phrase generated by the edge π_i. This approach can be generalized to find oracle hypotheses for higher-order approximation of BLEU score by first transforming the lattice so that each edge generates a n-gram instead of a word. However, for simplicity, we have only considered BLEU-1 approximation in our experiments.

Finally, solving the "loss-augmented" problem of Equation (7) can be done by defining the cost of an edge as the sum of the cost considered by the decoder and the cost considered by the oracle decoder.

In practice, to keep our implementation simple, we chose to rely on an external decoder to produce the lattices: before optimization, the whole training set is decoded using the same initialization as MERT and all the lattices are saved. Preliminary experiments show that this initialization has limited impact as long as the initial values of the weights are not unbalanced (i.e. no weight is set to 0 or a to a large value). Optimization is then performed, as described in Algorithm 1. As for MERT, the lattices can be regenerated occasionally, to make sure that they still represent an accurate approximation of the search space. However, experiments summarized in the next section show that it is sufficient for lattices to be regenerated only once.

An advantage of this implementation is that it can be used with *any* SMT system. Another way to proceed would be to decode and generate the lattice for sample s_i on an as-needed basis, i.e. upon updating the parameter value based on this particular sentence. While this solution might hasten convergence, it would require a tighter integration with the decoder and also more engineering work to avoid launching the decoder for each example.

5 EXPERIMENTS

We now describe the experiments made to validate our approach. Recall that our main motivation is to provide a much faster in-place replace-

ment of MERT: we are mainly interested in learning time and have only considered small and standard feature sets.

5.1 *Experimental Setup*

Two data sets were considered: the TED-talk English to French data set provided by the IWSLT'11 evaluation campaign and the French to English Europarl data set provided by the WMT'11 campaign. In all our experiments, we used the Moses decoder.

The TED-talk data set is a small data set made of a monolingual corpus ($111,431$ sentences) used to train the language model and a bilingual corpus ($107,268$ sentences) used to extract the phrase table. The Europarl system is trained using the parallel EPPS and News Commentary corpora ($1,940,639$ sentences). The target side of these two corpora were used to estimate a 4-gram language model with KN-smoothing.

For the TED-talk task, we used `dev-2010` dataset for training and `test-2010` for evaluation; the Europarl system was tuned on the dataset `test-2009` and evaluated on `test-2010`. Training for TED-talk task took 11 decodings of the training set (a wall time of almost 4 hours[1]) of the training set and achieved a BLEU score of 26.12 on the training set an of 23.28 on the test set; for Europarl, training took 10 decodings (more than 6 hours) and achieved a BLEU score of 21.47 on the training set and of 21.10 on the test set.

All reported BLEU scores were computed using the `multi-bleu` tool provided by Moses.As explained above, lattices are regenerated only once, after 300 iterations. Results fall down by about 2 BLEU points when the lattice are not regenerated, but regenerating the lattices more often did not yield any improvement.

5.2 *Learning Speed*

We first analyze the performance of the optimization procedure introduced in Section 3 by studying the evolution of the structured loss during optimization. Recall that the structured loss is a convex upper bound of (an approximation of) the BLEU score which defines the objective function optimized during training.

[1] All experiments are run on a single core of a server with 64G of RAM and 2 Xeon CPUs with 4 cores at 2.3 GHz. All reported times are wall time and include data loading.

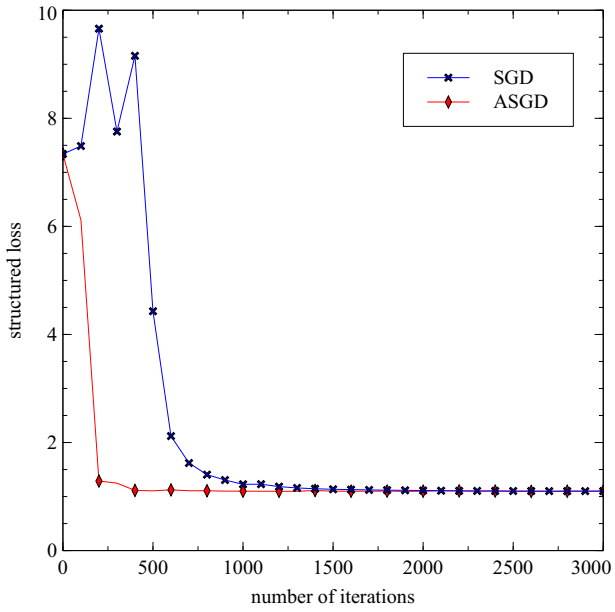


Fig. 1. Convergence of the (sub)gradient descent: evolution of the loss on TED-talk training set

Figure 1 represents the structured loss on the train set of the TED-talk task for two optimization strategies: plain stochastic gradient descent (SGD) and averaged stochastic gradient descent (ASGD). In both case, the learning rate has been set, according to the recommendations of [15]. It clearly appears that the neighborhood of the optimum is reached very quickly: for ASGD, Algorithm 1 converges after having seen only a few hundred examples. However, after reaching the optimum neighborhood, the weight vector is still changing and the objective function continues to decrease, albeit very slowly: the difference between two successive values after $1,000$ iterations is still in the order of 10^{-3}, which is much larger than the stopping criteria that are usually used. A difference in the order of 10^{-6} is only reached after $6,000$ iterations. Similar observations were made on the Europarl data.

To understand why convergence is fast, we have represented in Figure 2 the cosine similarity between the gradients of two examples of the training set after the first iteration of Algorithm 1. It appears that most

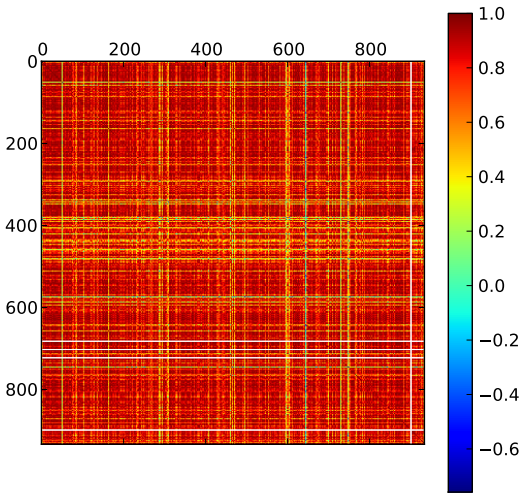


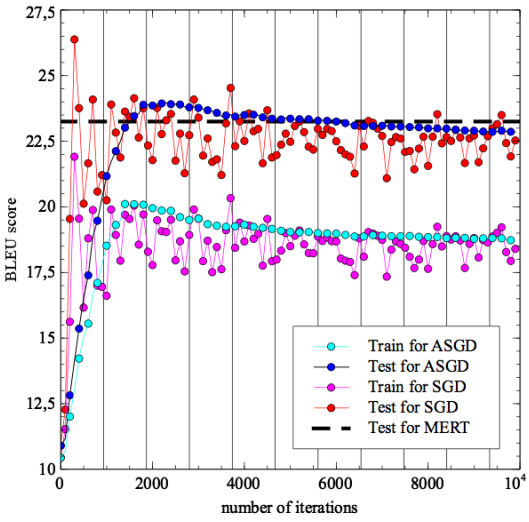
Fig. 2. Cosine similarity between the gradient of the examples in the TED-talk training set (most pairs show high similarity; lighter areas correspond to values in the middle of the scale)

gradients are very similar. This implies that the update in the online setting (based on a single example) is close to the update in the batch setting (after all examples have been seen), and that an online update, which requires N times less computation than a classical gradient update, will give (almost) the same results.

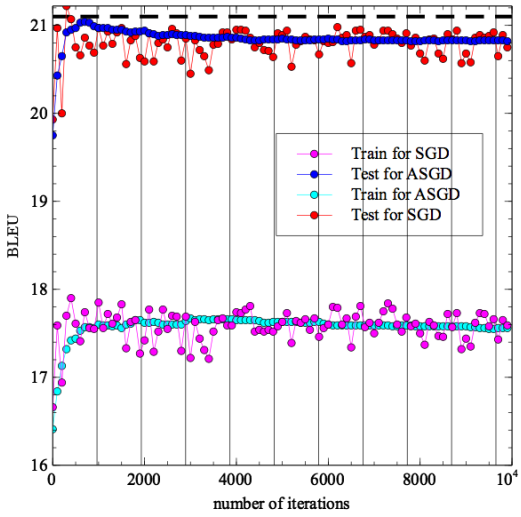
5.3 *Evolution of the BLEU Score*

As shown in the previous section, our optimization method is able to find the optimum of the learning criterion very quickly. However, this criterion is only an approximation of the BLEU score used to evaluate translation quality. In this section we study the quality of this approximation.

Figure 3 represents the evolution of the BLEU score on our two corpora. For SGD, the BLEU score on both the training set and the test set keeps changing during optimization: on the TED-talk training set, after $1,000$ iterations, the amplitude of the variations is still of several BLEU points even though the structured loss is almost stabilized. For ASGD,



(a) TED-talk corpus



(b) Europarl corpus

Fig. 3. BLEU scores on TED-talk and Europarl corpora. The dashed horizontal lines correspond to the score on the test set achieved by MERT and the vertical lines indicate iterations proportional to train set sizes.

Table 1. Comparison of MERT and of our approach (using binarized models).

	method	BLEU	# decodings of training set	training time (+ time to generate lattices)
TED-talk	MERT	23.28	11	3h39
	online	23.98	1.3	3mn (+ 5mn25)
Europarl	MERT	21.10	10	5h25
	online	21.04	1.3	6mn34 (+ 7mn30)

the regularization of the weight vector that results from its averaging over time reduces significantly the fluctuations of the BLEU scores. Nevertheless, for the two tasks, the trend is the same: at the beginning, performance quickly improves during the first few hundred iterations and then decreases slowly. Also note that *i)* the lattices have been regenerated only once during the optimization and that *ii)* the optimum BLEU value is reached, depending on the task, after $1,000$ or $2,000$ iterations. The corresponding total learning time is less than a few minutes with our simple and non-optimized implementation in Python. Table 1 summarizes the performances achieved by our approach and traditional MERT training.

In both cases, the observed variations of BLEU indicate that the upper bound used during optimization is not tight, which results from one of the following reasons: *i)* the way the optimization problem is convexified, *ii)* our sentence-level approximation of BLEU or *iii)* the additional approximations made when solving the loss-augmented or oracle decoding problems. To find out the source of the observed discrepancy, we have represented, in Figure 4, the evolution of both the BLEU-4 score used to evaluate translation quality and the average over the whole training set of the sentence-level BLEU-1 used during optimization. While these two scores are initially correlated (both of them are steadily increasing), this correlation seems to weaken with the number of iterations and, at the end, the BLEU-4 score is decreasing even if the BLEU-1 approximation continues to grow. Further experiments are still required to understand if this problem is the only responsible for the evolution of the BLEU-4 score during optimization.

5.4 *Stopping Criterion*

As shown in Figure 3, upon converging, our learning method is slightly outperformed by traditional MERT training. However, some of the weight

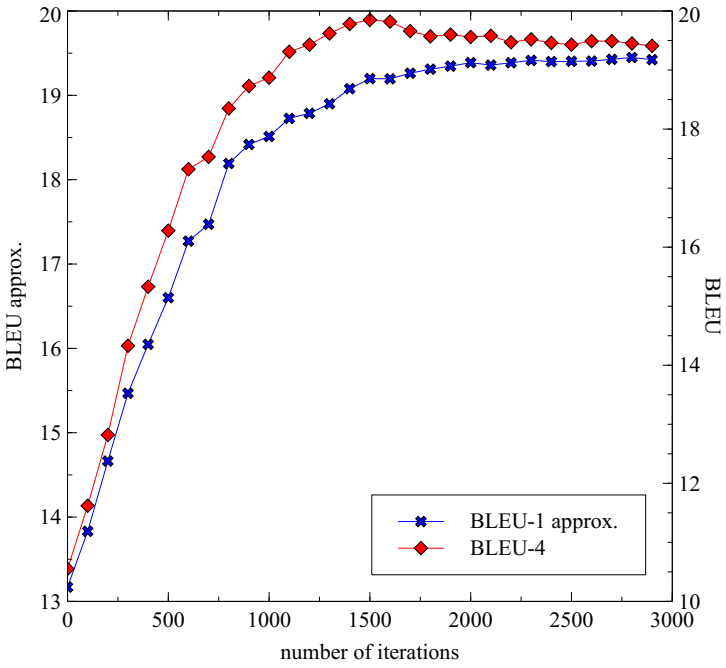


Fig. 4. BLEU-4 and BLEU-1 approximation during optimization on TED-talk for ASGD.

vectors found during optimization achieved better performance. Our approach is, therefore, only useful if we can find a criteria for stopping the optimization when a "good" weight vector is found. Fortunately, in all our experiments, we found that the BLEU scores on the training and on the test sets are highly correlated: their Pearson correlation coefficient is more than 0.92. The point that achieves an optimal BLEU score on the test set can therefore be easily identified by computing BLEU scores on the training set, which is done efficiently using a shortest path algorithm in the lattices without decoding the data again.

For the TED-talk task, the best point found by this method and the ASGD strategy slightly outperforms MERT by 0.7 points, while on the Europarl task MERT is better by 0.06 points. Using the SGD strategy leads to larger improvements at the expense of a higher variability in the score on the test set.

6 RELATED WORK

This paper is inspired by recent works on using structure learning techniques for SMT. This trend was pioneered by [18], who proposed to use a structured perceptron to train a PBSMT system. Like [6, 7], our approach augments the simple perceptron loss with a margin term. We however depart from these implementations in several ways. A first important difference is the use of an alternative optimization strategy, which, contrarily to the existing implementations of MIRA for MT, is really online and updates parameters after processing each instance. This is motivated by the observations of Section 5.2 and significantly speeds up learning. Another important difference is the use of lattices..

There are a number of additional small differences from MIRA, such as the approximation of the BLEU score, and the specific choice of the pseudo-reference: while the policy advocated in [7] selects a hypothesis that has both a high BLEU score and a good model score, our approach simply looks at BLEU scores. Incidentally, this difference makes our loss function slightly different from the one used in [7], as our pseudo-references are less dependent on the current value of the parameters. Altogether, it seems fair to state that our approach is conceptually much simpler to understand, to implement and to reproduce than approaches inspired by MIRA, which rely on the setting of many parameters such as the size of the n-best list, the slack parameter, the selection strategy for oracle hypotheses and their number, etc.

7 CONCLUSION

Building on recent advances in stochastic optimization and online machine learning, we have presented in this work an optimization method for the training of SMT systems. Our method achieved results that are at least as good as traditional MERT training, while being much faster. Another advantage of this technique is that it is based on the optimization of a convex objective function, implying that the resulting optimum will be less subject to variations, even in the presence of large feature sets.

While the performance obtained with a simple and straightforward implementation are already good, several questions remain open. We are, in particular, interested in understanding the impact of lattice sizes and of considering more features. Our future work will include a truly online implementation of this learning method within an open source decoder as well as a head to head comparison with MIRA.

ACKNOWLEDGMENTS This work has been partly financed by OSEO, the French State Agency for Innovation, under the Quaero program.

REFERENCES

1. Och, F.J.: Minimum error rate training in SMT. In: Proc. ACL'03, Sapporo, Japan (2003) 160–167
2. Foster, G., Kuhn, R.: Stabilizing minimum error rate training. In: Proc. WMT, Athens, Greece (2009) 242–249
3. Chatterjee, S., Cancedda, N.: Minimum error rate training by sampling the translation lattice. In: EMNLP'10, Stroudsburg, PA, USA, ACL (2010) 606–615
4. Kumar, S., Macherey, W., Dyer, C., Och, F.: Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In: Proc. ACL'09. (2009) 163–171
5. Smith, N.A.: Linguistic Structure Prediction. Synthesis Lectures on Human Language Technologies. Morgan and Claypool (May 2011)
6. Watanabe, T., Suzuki, J., Tsukada, H., Isozaki, H.: Online large-margin training for statistical machine translation. In: Proc. EMNLP'07, Prague, Czech Republic (June 2007) 764–773
7. Chiang, D., Marton, Y., Resnik, P.: Online large-margin training of syntactic and structural translation features. In: EMNLP'08. (2008)
8. Hopkins, M., May, J.: Tuning as ranking. In: EMNLP'11, Edinburgh, Scotland, UK., ACL (July 2011) 1352–1362
9. Taskar, B., Guestrin, C., Koller, D.: Max-margin Markov networks. In: NIPS 16. MIT Press, Cambridge, MA (2004)
10. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. JMLR **6** (December 2005) 1453–1484
11. Ratliff, N., Bagnell, J.A., Zinkevich, M.: (online) subgradient methods for structured prediction. In: Artificial Intelligence and Statistics. (2007)
12. Shor, N.Z.: Minimization Methods for Non-differentiable Functions. Springer-Verlag (1985)
13. Bertsekas, D.P. In: Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization. MIT Press (2012) 85–119
14. Bottou, L.: Online algorithms and stochastic approximations. In Saad, D., ed.: Online Learning and Neural Networks. Cambridge University Press, Cambridge, UK (1998)
15. Xu, W.: Towards optimal one pass large scale learning with averaged stochastic gradient descent. CoRR **abs/1107.2490** (2011)
16. Bach, F., Moulines, E.: Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In: NIPS 24. (2011)

17. Sokolov, A., Wisniewski, G., Yvon, F.: Computing lattice bleu oracle scores for machine translation. In: EACL'12, Avignon, France, ACL (April 2012) 120–129
18. Liang, P., Bouchard-Côté, A., Klein, D., Taskar, B.: An end-to-end discriminative approach to machine translation. In: ACL, Sydney, Australia (2006) 761–768

GUILLAUME WISNIEWSKI
LIMSI—CNRS,
UNIV. PARIS SUD,
91403 ORSAY CEDEX, FRANCE
E-MAIL: <WISNIEWS@LIMSI.FR>

FRANÇOIS YVON
LIMSI—CNRS,
UNIV. PARIS SUD,
91403 ORSAY CEDEX, FRANCE
E-MAIL: <YVON@LIMSI.FR>