Optimized MT Online Learning in Computer Assisted Translation

Prashant Mathur prashant@fbk.eu
FBK, Trento, Italy
DISI, University of Trento, Italy

Mauro Cettolo cettolo@fbk.eu
FBK, Trento, Italy

Abstract

In this paper we propose a cascading framework for optimizing online learning in machine translation for a computer assisted translation scenario. With the use of online learning, several hyperparameters associated with the learning algorithm are introduced. The number of iterations of online learning can affect the translation quality as well. We discuss these issues and propose a few approaches to optimize the hyperparameters and to find the number of iterations required for online learning. We experimentally show that optimizing hyperparameters and number of iterations in online learning yields consistent improvement against baseline results.

1 Introduction

The growing need of globalization has given a boost to the translation and localization industry where the high quality is guaranteed by human translators. To increase the productivity of these translators, Computer Assisted Translation (CAT) tools are used which provide access to translation memories, terminology, built-in spell checkers, dictionaries. A translation memory is a good source of previously translated segments; however, for new translation tasks, they are often obsolete. Due to the generalization capability of Machine Translation (MT) systems, they are employed in the back end of the CAT tools, for providing translation suggestions to the humans in cases where the translation memory fails. In fact, a seamless integration of SMT engines in CAT has shown to increase translator's productivity (Federico et al., 2012).

In state-of-the-art CAT tools, the SMT systems are generally static in nature and cannot adapt to the corrections posted by the translators. On the contrary, an adaptable SMT system would be preferable which can learn from the corrections of the post editor and modifies the statistical models accordingly. The task of learning from user corrections at the sentence level fits well in the online learning scenario. Online learning is a machine learning task where a predictor iteratively: (1) receives an input, (2) predicts an output label, (3) receives the correct output label from a human and, if the two labels do not match, (4) learns from the mistake.

However, the introduction of online learning itself brings two main issues. The first regards the tuning of the rate of learning, which is typically determined by the value of a number of parameters of the algorithm, hereafter referred to as hyperparameters[1]; optimizing them is then the first issue. The second problem is the selection of the optimal number of iterations of the online learning algorithm, i.e. an optimal stopping criterion.

In this paper, we focus on these issues and propose solutions in the context of SMT and CAT. Our work is an extension of Mathur et al. (2013), where three different hyperparameters are considered. Here, we are going to investigate techniques for optimizing the same , but in principle this work could be applied to any arbitrary number of hyperparameters.

[1] They are so called to distinguish them from the parameters of the models under analysis.

The organization of the paper is as follows. Section 2 gives an insight on the background needed to understand the concepts in the paper. Sections 3 and 4 describe the approaches we use to enhance the performance of the adaptable SMT system. Section 5 and 6 present experiments and results, respectively. Section 7 mentions a few related works and is followed by the conclusions section.

2 Background

Mathur et al. (2013) described an online learning approach for SMT integrated in CAT. In that paper, a twofold adaptation is proposed: 1) feature adaptation, in which an additional feature is added to the phrase table for rewarding the recently seen phrase pairs; 2) weight adaptation, where the log-linear interpolation weights of SMT model are adapted on the fly using MIRA (Watanabe et al., 2007). The online learning in the CAT framework is performed on the pair of source sentence and post edit, once the latter is provided by the human translator. From the implementation point of view, a particular structure where the source sentence is paired with its post edited translation is passed to the decoder: this activates a single online learning iteration. To perform multiple iterations, a corresponding number of copies of that structure has to be passed as input to the decoder.

The aforementioned paper does not deeply investigate the tuning of free parameters involved in that online learning process. In fact, hyperparameters are optimized by means of the Simplex algorithm, but the same values are then re-used for any possible number of iterations of the online learning, disregarding the dependence between the number of iterations and the hyperparameters, which is not a good assumption, as we will see later.

We extend that investigation from two viewpoints. First, we focus on the selection of better hyperparameters; second, we look for the optimal number of iterations of online learning required to maximize the performance of an adaptable SMT system.

3 Optimization

The following steps lay the optimization process in a cascaded framework:

1. Baseline SMT models are tuned on the development set.
2. Copies of development set are made such that each copy represents a different number (i) of iterations of online learning ($i \in 1..10$).
3. The tuned log-linear weights are kept fixed and hyperparameters are tuned by derivate free optimization (DFO) methods.

The optimal weights are computed by minimizing the error on a held-out parallel development set by means of MIRA which operates on the N-best list and re-ranks it by changing the log-linear weights. Since hyperparameters do not affect this N-best list once it is created, there is no direct way to optimize the hyperparameters on the development set via traditional tuning methods. An alternative solution needs to be found.

Hyperparameters in SMT models, such as *distortion limit* and *beam size*, have been typically optimized using derivative free optimization (DFO) techniques such as Simplex (Chung and Galley, 2012) and Hill Climbing (Stymne et al., 2013). Analogously, once we have tuned the log-linear weights, we keep them fixed and optimize our hyperparameters by means of DFO. This cascade approach prevents joint optimization over 18 parameters[2] which is not feasible using the DFO techniques because they tend not to converge with so many parameters.

In this paper we focus on three hyperparameters, namely the *feature learning rate* (FLR), the *weight learning rate* (WLR) and the *slack variable* (SLK). FLR determines the rate of learning of the additional online feature; WLR and SLK control respectively the learning rate and the size of the update of the online learning algorithm (MIRA) employed for updating the log-linear weights. Their optimization is performed with respect to a loss function defined over an objective MT evaluation metric, by the two DFO techniques described in the following.

[2] 14 weights from SMT models, plus 1 additional weight for the online learning feature and 3 hyperparameters.

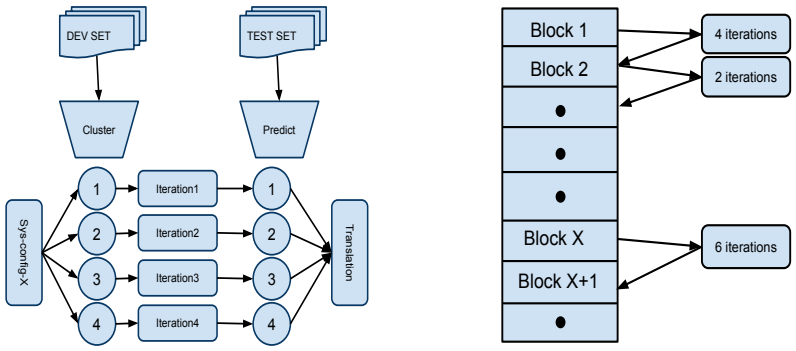

Figure 1: Clustering (left) / blockwise (right) approaches to find the optimal number of iterations for online learning.

Downhill Simplex Method The Downhill Simplex method, also known as Nelder-Mead method (Nelder and Mead, 1965), is a technique for minimizing an objective multivariate function. The method is iterative and approximates a local optimum by using a simplex, that is a polytope of $N + 1$ vertices in N dimensions. At each iteration, a new test position is evaluated by extrapolating the behavior of the objective function measured at each vertex of the simplex. The algorithm then chooses to replace one of the vertices with the new test point and so the search progresses. New test positions are generated so that the simplex is stretched along promising lines (when the simplex is still far from any optimum) or shrinked towards a better point (when it is close to a local optimum).

Modified Hill Climbing Hill Climbing is generally used for a single variate function $f(x)$: it fluctuates the value of the variable x and computes the loss incurred by the function $f(x)$. Step by step, the method moves the variable towards the direction where the loss incurred is minimum. We extended the same optimization to multivariate functions $f(x_1, x_2, \ldots, x_n)$ by moving one variable at a time. Moreover, we modified the original hill climbing by initially allowing the variable to take large steps in the convex space, and then constrain the variable to take smaller steps, similar to simulated annealing (Kirkpatrick et al., 1983). This allows Hill Climbing to converge faster than in the standard approach.

Later, we will see that the optimal value of hyperparameters depends on the number of iterations used for the online learning; therefore, the cascade optimization process is run ten times with different numbers i of iterations ($i \in 1..10$). Given that hyperparameters are optimized with two derivative free optimizers, a total of twenty different optimal configurations are available for each SMT system to test.

4 Stopping criteria

Once the optimal values of the log-linear weights and of hyperparameters have been estimated, next step towards improving the online learning is to find the optimum number of iterations required to learn. The model resulting from the run of the optimum number of iterations should ideally outperform the models obtained with a random iteration number and avoid overfitting on the data. We propose two solutions to find this optimum number: one, named `clustering`, needs a pre-processing step on the development set; the other, named `blockwise`, works on the evaluation data directly.

Clustering In the clustering approach, as a pre-processing stage, the development set is partitioned into k clusters and the optimal number of iterations for each cluster is determined. At run-time, each test sentence is classified into one of the clusters and the corresponding optimal number of iterations is used for the online learning for that source sentence and post-edit. The k-means clustering with random seeding is performed to cluster the development set, using the

Euclidean distance as similarity metric;[3] bilingual development sentences are clustered on the basis of the entropy of both source and target sides. Once the development set is clustered, the optimal number of iterations for each cluster is computed as follows: online learning is run on each cluster for $i \in \{1 \dots 10\}$ iterations, keeping track of the error rate at each iteration; each cluster is then associated with the iteration number corresponding to the minimum error rate. The scheme on the left of Figure 1 represents the clustering approach.

Blockwise In the blockwise approach, the test set is split into blocks of $N \in \{10, 20, 30\}$ sentences.[4] Once the X^{th} block has been post-edited by the user, the optimal number of iterations (O_X) for that block is found, by comparing the error rates yielded by running the online learning for i iterations and selecting the best performing iteration number. O_X is then used to perform the online learning on each segment (and post-edit) of the $X + 1^{th}$ block, till the whole block is processed. The blockwise method is depicted on the right side of Figure 1.

5 Experimental Setup and Preliminary Analysis

5.1 Data

We evaluated our methods for optimizing the online learning algorithm on three translation tasks defined over two domains, namely Information Technology (IT) and Legal. The IT test set is proprietary, involves the translation of technical documents from English into Italian and has been used in the field test recently carried out under the MateCat project.[5] In the Legal domain, experiments involved the translation of English documents into either Spanish or French; training and evaluation sets belong to the JRC-Acquis corpus (Steinberger et al., 2006) so that the effectiveness of the proposed approaches is assessed also on publicly available data.

Since our methods regard the adaptation of MT models, the potential impact strictly depends on how much the considered text is repetitive. For measuring that text feature, we use the repetition rate proposed by Bertoldi et al. (2013). Statistics of the parallel sets of both source and target side along with the repetition rates are reported in Table 1.

Domain	Set	#srcTok	srcRR	#tgtTok	tgtRR
$\text{IT}_{\texttt{en}\rightarrow\texttt{it}}$	Train	57M	na	60M	na
	Dev	3.7K	7.65	4K	7.61
	Test	3.4K	34.33	3.7K	33.90
$\text{Legal}_{\texttt{en}\rightarrow\texttt{es}}$	Train	56M	na	62M	na
	Dev	3K	24.09	3.5K	24.47
	Test	11K	20.67	12.5K	20.07
$\text{Legal}_{\texttt{en}\rightarrow\texttt{fr}}$	Train	63M	na	70M	na
	Dev	3K	23.52	3.7K	23.42
	Test	11K	20.67	13K	20.92

Table 1: Statistics of the parallel data along with the corresponding repetition rate (RR).

5.2 Reference Systems

The SMT systems are built using the Moses toolkit (Koehn et al., 2007). Domain specific training data is used to create translation and lexical reordering models. 5-gram language models for each task, smoothed through the improved Kneser-Ney technique (Chen and Goodman, 1998), are estimated by means of the IRSTLM toolkit (Federico et al., 2008) on the target side of the training parallel corpora. The weights of the log-linear interpolation of MT models are optimized using the MIRA (Watanabe et al., 2007) implementation provided in the Moses toolkit.

[3] The Cosine distance was also tested: it performed similarly to the Euclidean distance, but Euclidean distance gave better quality of clusters than Cosine.

[4] In real texts, we can assume that bunches of some tens of segments (e.g. 10-30) are linguistically coherent such that an adaptation scheme can be effectively applied.

[5] http://www.matecat.com

Performance is reported in terms of BLEU (Papineni et al., 2002) and TER (Snover et al., 2006). Details on the tested SMT systems follow:

Baseline The static baseline system does not perform any online learning, hence there are no hyperparameters involved in the system.

Def-Param-*x Online learning systems using default values of hyperparameters and running 1, 5 and 10 iterations of online learning. These systems provide a reference for assessing the usefulness of estimation of the optimal number of iterations vs. the use of a pre-defined number of iterations. The default values of the hyperparameters are FLR=0.1, WLR=0.05 and SLK=0.001, which yield reasonable performance in preliminary investigations.

5.3 Optimization of the Hyperparameters

Having tuned the log-linear weights, the following systems are built:

Opt-Param-*x Online learning systems with hyperparameters optimized by means of either Simplex or Hill Climbing techniques of Section 3.

Figure 2 shows the convergence of the DFO methods over their number of iterations[6], while keeping fixed the number of iterations of the online learning (just one, i.e. $1\times$) and the log-linear weights.

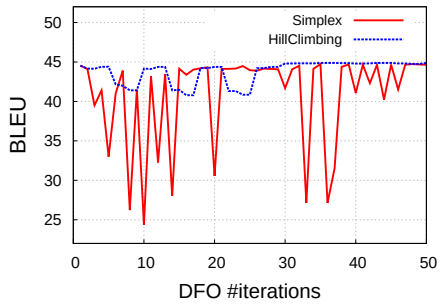

Figure 2: Simplex vs. Modified Hill Climber on the Legal/en→fr test set, with $1\times$ iteration of the online learning algorithm.

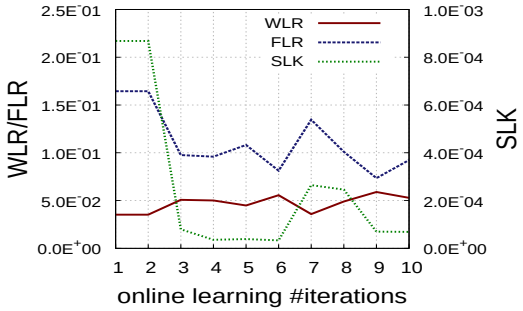

Figure 3: Optimal values of hyperparameters computed by the Simplex for different iteration numbers of the online learning algorithm (IT/en→it task).

From the figure, we can argue that the Simplex attempts many parameter values performing quite differently, and finally converges to an optimum. Hill Climbing converges to the optimum faster, but on the other side it seems to explore a smaller portion of the search space. For assessing the guess, we tried to change the starting seeds of the two optimizers: Simplex still found the same optimum, while Hill Climbing failed even with 50 iterations, confirming our intuition. For this reason, in Opt-Param-*x systems we are only going to provide results obtained using hyperparameters optimized via Simplex.

Figure 3 shows the optimal value of hyperparameters as a function of the number of iterations of the online learning algorithm. Apart a general and reasonable tendency to define smaller updates when more iterations are performed, it is worth to note that the optimal hyperparameter values do change with the number of iterations, again confirming our intuition.

As described in Section 3, tuning of hyperparameters was performed for different numbers of iterations of online learning, resulting in 10 different configurations for each DFO algorithm.

5.4 Online Learning Stopping Criteria

At the end of the optimization stage, 10 optimal configurations for each of the two DFO techniques are available for testing. In both DFOs, a TER-based loss function is employed, since

[6]These are the iterations required by Simplex/HillClimbing to converge.

clusters/blocks can be too small to allow the reliable computation of BLEU values. A total of 20 optimized systems are then run to look for the optimal number of iterations of the online learning to be used on the test sets: this optimal number is found on the development set with the clustering technique, directly on the test set with the blockwise technique.

Clustering As already mentioned, we first partition the development set using k-means clustering, where k takes values in $\{2, 4, 6, 8, 10, 12\}$. In principle, we can increase k but that would decrease the size of the clusters with the risk of data overfitting. The development set of the IT task consists of 300 sentences, i.e. 300 data points, hence for consistency purposes we set the maximum value of k to $\left\lfloor \sqrt[2]{300/2} \right\rfloor = 12$ for all tasks.[7]

For each cluster, we pick the configuration[8] which performs best on the development set; the test sentences that are assigned to that cluster are then translated with the chosen optimal configuration.

Figure 4 reports the average number of iterations of online learning required by the clustering technique to converge for different values of k. It is shown that the larger the number of clusters, the faster the convergence of the online learning algorithm; this is because the online learner has less to learn from small clusters, even performing more and more iterations.

Blockwise In the blockwise approach the number of iterations is optimized directly (but fair) on the test set. The optimal number for a given block is found once its post-edits are available and is used for the translation of the following block; this step is iterated for all blocks. In other words, number of iteration of online learning on the current block is decided by optimizing the number of iterations on the previous block. For the first block of the test set, the optimal configuration on the development set is considered.

Anyway, two main issues arise: 1) which system configuration should we use for the first block? 2) what should be the size of the block? We decided on the following. First, the optimal number of iterations for a block is found by comparing the 10×2 configurations optimized on the development set. This is analogous to what we do in the clustering approach.

Secondly, for testing their effect on the number of iterations, different sizes of the block $(10, 20, 30$ sentences) are tried. Figure 5 shows how the block size affects the optimal number of online learning iterations for one of the considered tasks.

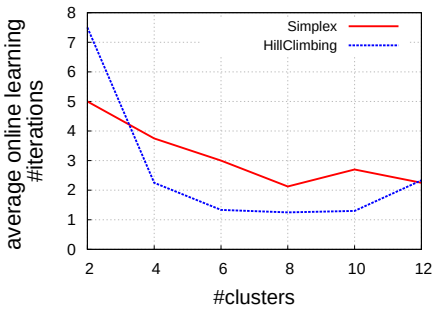

Figure 4: Average number of iterations of the online learning algorithm per number of clusters for the two optimizers (Legal/en→fr task).

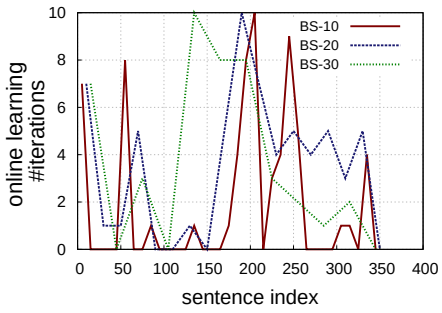

Figure 5: Effect of varying the block size on the number of iterations of the online learning algorithm (IT/en→it).

[7] Rule of thumb according to Mardia et al. (1980).

[8] These configurations are not compared all together at once; instead, we separately compare the 10 configurations for each DFO method.

6 Results

Reference Table 2 collects results from the baseline and online learning systems mentioned in Sections 5.2 and 5.3.

System	IT en→it		Legal en→fr		Legal en→es	
	BLEU	TER	BLEU	TER	BLEU	TER
Baseline	46.73	31.97	33.69	51.49	35.65	50.04
Def-Param-1x	46.27	31.23	34.28	50.31	35.28	48.07
Def-Param-5x	42.61	34.90	33.04	51.51	32.13	51.12
Def-Param-10x	39.18	37.66	34.34	50.25	31.08	52.74
Opt-Param-1x	46.56	31.41	34.24	50.34	35.38	48.34
Opt-Param-5x	44.48	33.28	33.32	50.87	32.68	50.82
Opt-Param-10x	47.11	31.41	34.25	50.47	34.61	48.78

Table 2: MT scores for all tasks of the following systems: baseline; online learning with default values of hyperparameters; online learning with optimized values of hyperparameters by means of Simplex. Online learning is performed for fixed numbers of iterations (1,5,10).

The online learning system with $10\times$ iterations and optimized hyperparameters outperforms the baseline by 0.5 to 1 BLEU/TER points on both IT/en→it and Legal/en→fr tasks. On the Legal/en→es task, the best performance is obtained by performing 1 iteration of the online learning, that allows to clearly beat the baseline by 1.70 TER points (48.34 vs. 50.04).

In two out of three tasks (en→it and en→es), the performance of the systems with default parameters decreases rapidly as the number of iterations increases, because the parameters are not tuned properly on the held-out dev set. This confirms our assumption that the value of hyperparameters plays an important role on system performance. In the Legal en→fr system, incidently the default value of hyperparameters is close to the optimized one and hence performance are pretty similar in the two setups ([Def|Opt]-Param-*x), better than the baseline by around 0.5 BLEU points and 1 TER point when online learning is iterated either 1 or 10 times.

Table 3 collects results of systems with hyperparameters optimized on the basis of the optimal number of online learning iterations, as determined by means of the two investigated techniques (blockwise and clustering). As a first general consideration, apart few exceptions, Simplex is more effective than Hill Climbing in optimizing hyperparameters; therefore, in the following discussion, we focus on it.

stopping technique setup		IT en→it				Legal en→fr				Legal en→es			
		simplex		hill climbing		simplex		hill climbing		simplex		hill climbing	
		BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
block size	10	46.80	31.43	46.31	31.48	34.64	50.18	33.68	49.95	34.99	48.87	34.16	49.31
	20	46.30	31.81	46.11	31.96	34.98	49.63	34.47	49.66	35.64	48.33	34.44	48.70
	30	45.42	32.42	46.40	31.71	34.78	50.68	34.30	51.13	35.68	48.67	34.63	48.64
number of clusters	2	46.80	30.95	46.33	31.28	34.90	50.75	35.02	50.80	36.13	47.78	36.30	47.82
	4	45.99	32.09	46.07	31.33	35.09	50.24	35.24	50.71	36.05	47.89	36.05	47.74
	6	46.41	31.08	46.06	31.48	34.66	50.52	34.40	50.59	35.71	48.07	35.86	47.77
	8	46.03	31.81	45.76	31.68	35.07	50.58	35.06	50.78	35.75	48.23	36.11	47.76
	10	44.98	32.67	46.32	31.31	35.12	50.62	35.23	50.94	36.08	47.56	35.69	47.87
	12	46.23	31.74	46.79	30.77	35.15	50.62	35.07	50.74	36.34	47.74	35.58	47.93

Table 3: Results of the blockwise/clustering techniques on the three considered tasks, by varying the block size/number of clusters. Performance of the two DFO methods is reported.

Blockwise The upper part of the table reports results employing the blockwise technique. On the IT/en→it task, the blockwise system (block size 10) outperforms the baseline in terms of TER and gives comparable performance to the optimized system (with 10x iterations). To note that this same quality is obtained more efficiently: in fact, since for each block no more than 10 iterations are performed (and less likely, according to Figure 5), on an average it is expected that the global cost on the whole test set is lower than the cost of the Opt-Param-10x system, where 10 iterations are performed on each sentence of the test set.

On the Legal/en→fr task, the blockwise system (block size 20) outperforms the baseline as well as the best performing online learning system with fixed number of iterations (Def-Param-10x) by 1.86 and 0.62 TER points, respectively. However, the size of the block yielding the highest performance differs between the two domains; that is probably because the IT test set is a collection of different technical documents which makes it rather heterogeneous, while the Legal test sets, being from single documents, are much more homogeneous, allowing the use of larger blocks.

On the Legal/en→es task, the block size does not impact significantly on the performance of the online learning system; although there is no BLEU gain in comparison to the baseline system, TER improves by up to 1.71 points (48.33 vs. 50.04, when block size is set to 20).

Clustering Concerning the clustering technique, results are shown in the bottom part of Table 3. Similarly to the blockwise method, for the IT/en→it we do not see any BLEU gain, while TER improves the baseline by even more than 1 point (30.95 vs. 31.97).

For the Legal/en→fr task, an increase of about 1.5 BLEU points is observed for most of the cluster sizes. Even in terms of TER, the number of cluster (and then the cluster size) seems not to affect much the scores, which lower the baseline TER (51.49) by around 1 point.

A behavior similar to IT/en→it is observed in the Legal/en→es task: minimal impact of the cluster size, small BLEU improvements (no more than 0.7 points), larger TER gains (even more than 2 points).

Summarizing, we see consistent improvements of TER, but not of BLEU. A possible explanation is the use of TER as the error metric for finding the optimal iterations of online learning for the blocks and the clusters.In fact, the size of clusters is too small to allow the reliable computation of the BLEU, but optimizing TER favors short sentences, which lower BLEU through the brevity penalty.

7 Related Work

Online learning for SMT has emerged as a hot topic over the last decade (Nepveu et al., 2004; Hardt and Elming, 2010; Ortiz-Martínez et al., 2010; Martínez-Gómez et al., 2012; Denkowski et al., 2014). Nevertheless, to our knowledge, optimizing the number of iterations of online learning has not been previously studied in the context of SMT integrated in CAT tools. Therefore, this is the first work towards a fully optimized MT online learning system for CAT.

The most notable work in the field of optimization of hyperparameters in MT is that by Chung and Galley (2012) where the decoder is integrated with a minimizer so that they can optimize the values of free parameters such as beam size and distortion limit. This minimizer runs derivative free optimization techniques, such as Powell and Nelder-Mead methods, to optimize log-linear weights as well as the hyperparameters. They also argue that this integrated minimizer measures the *true error rate* whereas MERT minimizes the *artificial error rate* computed on a N-best list. Their use of DFO methods pushed us to adopt the same.

Later, Stymne et al. (2013) focused on using the distortion limit (DL) in the document level decoder Docent (Hardmeier et al., 2013). Their system provides better BLEU scores when there is a soft constraint on the DL (i.e. DL is tunable) rather than a hard constraint (DL not tunable). This experiment further supports the optimization of MT parameters in order to gain performance.

Learning of hyperparameters has been a widely studied topic in machine learning. Grid search, random search (Bergstra and Bengio, 2012), Gaussian process (Snoek et al., 2012) are only a few methods that have been used in the past for hyperparameter optimization. Gradient-

based hyperparameter learning algorithms have been proposed for a variety of supervised learning models such as neural networks (Larsen et al., 1996). In our case, the evaluation of the loss function is a costly procedure which requires the translation of the whole development set: the application of the above approaches can then be unfeasible unless we use *Racing* or *Lattice based decoding* (Chung and Galley, 2012).

8 Conclusion

We have shown that online learning can be effectively integrated into MT for CAT by following a cascaded framework where one first optimizes the extra parameters involved with the learning algorithm, and then find the optimal number of iterations of online learning required on the test set. We experimented with two derivative free optimization techniques, namely Simplex and Hill Climbing, and showed their convergence. Two techniques, Blockwise and Clustering, are instead proposed to find the optimal number of iterations. After an extensive set of experiments we can conclude that the clustering technique performs better than the blockwise one when the test set is homogeneous in nature, otherwise the blockwise with small blocks is preferable.

Acknowledgements

This work was supported by the MateCat project (grant agreement 287688), which is funded by the EC under the 7^{th} Framework Programme.

References

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Bertoldi, N., Cettolo, M., and Federico, M. (2013). Cache-based online adaptation for machine translation enhanced computer assisted translation. In *Proc. of MT Summit*, pp. 35–42, Nice, France.

Chen, S. F. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.

Chung, T. and Galley, M. (2012). Direct error rate minimization for statistical machine translation. In *Proc. of WMT*, pp. 468–479, Montréal, Canada.

Denkowski, M., Dyer, C., and Lavie, A. (2014). Learning from post-editing: Online model adaptation for statistical machine translation. In *Proc. of EACL*, pp. 395–404, Gothenburg, Sweden.

Federico, M., Bertoldi, N., and Cettolo, M. (2008). IRSTLM: An open source toolkit for handling large scale language models. In *Proc. of Interspeech*, pages 1618–1621, Brisbane, Australia.

Federico, M., Cattelan, A., and Trombetti, M. (2012). Measuring user productivity in machine translation enhanced computer assisted translation. In *Proc. of AMTA*, San Diego, US-CA.

Hardmeier, C., Stymne, S., Tiedemann, J., and Nivre, J. (2013). Docent: A document-level decoder for phrase-based statistical machine translation. In *Proc. of ACL: System Demonstrations*, pp. 193–198, Sofia, Bulgaria.

Hardt, D. and Elming, J. (2010). Incremental re-training for post-editing SMT. In *Proc. of AMTA*, Denver, US-CO.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL, Companion Volume of the Demo and Poster Sessions*, pp. 177–180, Prague, Czech Republic.

Larsen, J., Hansen, L. K., Svarer, C., and Ohlsson, M. (1996). Design and regularization of neural networks: The optimal use of a validation set. In *Proc. of IEEE Signal Processing Society Workshop*, pp. 62–71, Kyoto, Japan.

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1980). Multivariate analysis. Academic Press.

Martínez-Gómez, P., Sanchis-Trilles, G., and Casacuberta, F. (2012). Online adaptation strategies for statistical machine translation in post-editing scenarios. *Pattern Recogn.*, 45(9):3193–3203.

Mathur, P., Cettolo, M., and Federico, M. (2013). Online learning approaches in computer assisted translation. In *Proc. of WMT*, pp. 301–308, Sofia, Bulgaria.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4):308–313.

Nepveu, L., Lapalme, G., Langlais, P., and Foster, G. (2004). Adaptive language and translation models for interactive machine translation. In *Proc. of EMNLP*, pp. 190–197, Barcelona, Spain.

Ortiz-Martínez, D., García-Varea, I., and Casacuberta, F. (2010). Online learning for interactive statistical machine translation. In *Proc. of HLT-NAACL*, pp. 546–554, Los Angeles, US-CA.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *Proc. of ACL*, pp. 311–318, Philadelphia, US-PA.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Proc. of NIPS*, pp. 2951–2959, Lake Tahoe, US-NV.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proc. of AMTA*, pp 223–231, Boston, US-MA.

Steinberger, R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufiş, D., and Varga, D. (2006). The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proc. of LREC*, pp. 2142–2147, Genoa, Italy.

Stymne, S., Hardmeier, C., Tiedemann, J., and Nivre, J. (2013). Tunable distortion limits and corpus cleaning for SMT. In *Proc. of WMT*, pp. 225–231, Sofia, Bulgaria.

Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. (2007). Online large-margin training for statistical machine translation. In *Proc. of EMNLP*, pp. 764–773, Prague, Czech Republic.