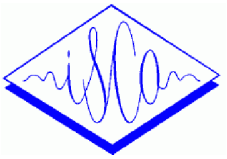
ISCA Archive
http://www.isca-speech.org/archive



International Workshop on Spoken
Language Translation (IWSLT) 2006
Keihanna Science City, Kyoto, Japan
November 27-28, 2006

Tuning Machine Translation Parameters with SPSA

Patrik Lambert, Rafael E. Banchs

TALP Research Center,
Jordi Girona Salgado 1–3. 08034 Barcelona, Spain
lambert,rbanchs@gps.tsc.upc.edu

Abstract

Most of statistical machine translation systems are combinations of various models, and tuning of the scaling factors is an important step. However, this optimisation problem is hard because the objective function has many local minima and the available algorithms cannot achieve a global optimum. Consequently, optimisations starting from different initial settings can converge to fairly different solutions. We present tuning experiments with the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm, and compare them to tuning with the widely used downhill simplex method. With IWSLT 2006 Chinese-English data, both methods showed similar performance, but SPSA was more robust to the choice of initial settings.

1. Introduction

Statistical machine translation (SMT) was originally based on the noisy channel approach [1]. In present SMT systems, the noisy channel approach has been expanded to a more general maximum entropy approach in which a log-linear combination of multiple feature functions is implemented [2]. Translation quality can be improved by adjusting the weight of each feature function in the log-linear combination. This can be effectively performed by minimising translation error over a development corpus for which manually translated references are available [3].

This minimisation problem in multiple dimensions is difficult because of three main characteristics of the objective function. Firstly, it has no analytic representation, so the gradient cannot be calculated. Secondly, it has many local minima. Finally, its evaluation has a significant computational cost (depending on the scheme, it implies translating the development corpus or re-ranking an n-best list for this corpus, and calculating some translation error measure). Gradient may be approximated, but this is costly since it requires typically as many function evaluations as the number of scaling factors. Thus, algorithms based on derivatives are discarded. Algorithms which require many objective functions evaluations, such as simulated annealing or genetic algorithms, are also discarded. Two popular alternatives are Powell's method [4, 5, 3] and the downhill simplex method [6, 5, 7]. In recent experiments at the 2006 John Hopkins University Summer Workshop on SMT, both methods achieved similar performance [8]. The simplex method is self-contained and straightforward and thus widely used for SMT tuning [7, 9, 10, 11], although it is not very efficient in terms of number of objective function evaluations for a high number of dimensions [12].

However, from the authors experience, a slight modification of initial parameters in the simplex optimisation can result in an appreciable difference in both the value of the local minimum found and the value of the optimal parameters. This difference is transmitted when these parameters are used to translate a test corpus. When a translation system is compared to a baseline, the difference arising only from the tuning process can be even greater than the difference arising from the two systems differences, leading to insignificant results. In some data sets, some inconsistencies of the tuning method have also been reported [13].

In this paper we compare tuning with the Downhill Simplex method and with the Simultaneous Perturbation Stochastic Approximation [14] (SPSA) method. SPSA has been successfully applied in areas including statistical parameter estimation, simulation-based optimisation, signal and image processing [15].

This paper is structured as follows. First the essential features of the SPSA method are presented. Then in section 3, objectives and details of the experimental work are given. In section 4, results are shown and discussed. Finally, some concluding remarks and perspectives are given.

2. Presentation of SPSA algorithm

The SPSA method is based on a gradient approximation which requires only two evaluations of the objective function, regardless of the dimension of the optimisation problem. This feature makes it especially powerful when the number of dimensions is increased.

The SPSA procedure is in the general recursive stochastic approximation form:

$$\hat{\lambda}_{k+1} = \hat{\lambda}_k - \mathbf{a}_k \hat{\mathbf{g}}_k(\hat{\lambda}_k) \quad (1)$$

where $\hat{\mathbf{g}}_k(\hat{\lambda}_k)$ is the estimate of the gradient $\mathbf{g}(\lambda) \equiv \partial E/\partial\lambda$ at the iterate $\hat{\lambda}_k$ based on the previous mentioned evaluations of the objective function. a_k denotes a positive number that usually gets smaller as k gets larger. Two-sided gradient approximations involve evaluations of $E(\hat{\lambda}_k + \text{perturbation})$

and $E(\hat{\lambda}_k - \text{perturbation})$. In the simultaneous perturbation approximation, all elements of $\hat{\lambda}_k$ are randomly perturbed together and the approximated gradient vector is:

$$\frac{E(\hat{\lambda}_k + c_k\mathbf{\Delta_k}) - E(\hat{\lambda}_k - c_k\mathbf{\Delta_k})}{2c_k}\begin{bmatrix}1/\Delta_{k1}\\1/\Delta_{k2}\\\vdots\\1/\Delta_{kN}\end{bmatrix} \quad (2)$$

In equation 2, $\mathbf{\Delta}_k$ is a perturbation vector of same dimension N as λ, whose values Δ_i are computed randomly. c_k denotes a small positive number that usually gets smaller as k gets larger. Compared to a finite-difference gradient approximation, involving N times more function evaluations, the simultaneous approximation causes deviations of the search path. These deviations are averaged out in reaching a solution and according to [15], under reasonably general conditions, both gradient approximations achieve the same level of statistical accuracy for a given number of iterations. Notice that in general, SPSA converges to a local minimum.

The general form of the algorithm consists of the following steps (see section 3.4 for further implementation details):

Step 1 Calculate gain sequences a_k and c_k.

Step 2 Generate the simultaneous perturbation vector $\mathbf{\Delta}_k$.

Step 3 Evaluate $E(\hat{\lambda}_k + c_k\mathbf{\Delta_k})$ and $E(\hat{\lambda}_k - c_k\mathbf{\Delta_k})$.

Step 4 Approximate the gradient as in equation 2

Step 5 Update λ estimate as in equation 1

Step 6 Iteration or termination. Return to Step 2 with $k+1$ replacing k. Terminate if the maximum number of iterations have been reached or if there is little change in several successive iterates.

3. Experimental Settings

3.1. Translation system used

Although the following discussion would be valid in many contexts, and in particular for any Empirical MT system, it is convenient here to present briefly the models implemented by our system, and whose respective weights are tuned. For a more complete description see [16]. The SMT approach used here considers a translation model which is based on a 4-grams language model of bilingual units which are referred to as tuples. Tuples are extracted from Viterbi alignments and can be formally defined as the set of shortest phrases that provides a monotonic segmentation of the bilingual corpus.

In addition to the bilingual 4-gram translation model, the translation system implements a log linear combination of five additional feature functions: a 4-gram language model of the target language (denoted TM); a 4-gram language model of target POS-tags (TTM) which helps, along with the target language model, to provide a better concatenation of tuples; a word bonus feature (WB), which compensates the system preference for short translations over large ones; and finally, two lexicon models (L1 and L2) that implement, for a given tuple, the IBM-1 translation probability estimate between the source and target (or target and source, respectively) sides of it.

3.2. Objectives

Thus we have a translation system whose outcome depends on a set of parameters λ (in this experiment, parameters were restricted to the scaling factors of the various models). We want to minimise a function $E(\lambda)$, which measures the translation errors over a given development set, made by the system with the parameter vector λ. In this experiment, each evaluation of $E(\lambda)$ implies computing full translation of the development corpus, which is computationally intensive (the number of evaluations of $E(\lambda)$ to achieve convergence is in the order of 100). Note that in other setups [3, 8], tuning is performed in two stages. In the first stage, full translation of the development corpus is computed, with n-best output. In the second stage, the n-best list is re-ranked. In this second stage, a parameter optimisation is performed with the downhill simplex method or with Powell's method [1]. In this case, an evaluation of the objective function only implies re-ranking the n-best list. The number of re-rankings necessary to achieve convergence is also in the order of 100. After this optimisation, the optimum parameters are used to translate again the development corpus and generate an updated n-best list. In this setup, convergence (on the first stage level) usually occurs after less than 10 full translations.

The objective of the experiment was to perform the optimisation of $E(\lambda)$ with the downhill simplex method and the SPSA method, and to compare the consistency of the results over changes in initial settings. For this, we ran the algorithms from 7 different initial points and for each point, for 10 slightly different realisations. For both algorithms, an evaluation of $E(\lambda)$ implied a translation of the development corpus by exactly the same system (except the model weights). Thus, an objective function evaluation had the same computational cost for both algorithms.

We aimed at choosing initial points well distributed in parameter space, but nevertheless realistic. Notice that in the log-linear combination, weights can be rescaled to set one of the parameters to some value, so the translation model was set to 1 and kept fixed during optimisation. In the first initial point, all parameters are also 1, so that all models start with equal weights. In the second initial point, all parameters are equal to 0.5. The other points were chosen in the following way. We collected sets of optimal parameters obtained previously on another development corpus, and noted down in which range the scaling factor of each model behaved. We selected the initial value of each parameter randomly within its corresponding range. Table 1 displays the initial points

[1] Actually, SPSA could also be used instead

used in the experiments.

ID	TM	TTM	WB	L1	L2
1	0.29	0.52	0.32	1.7	0.84
2	0.5	0.5	0.5	0.5	0.5
3	0.58	0.42	1.4	0.2	0.075
4	1	1	1	1	1
5	1.1	0.22	1.5	1.6	0.29
6	1.2	0.53	1.5	1.3	0.89
7	1.3	0.34	1.2	0.85	0.44

Table 1: Sets of initial parameters used in the experiments. In table 3 points are referred to by their ID number.

The error function we choose is the BLEU score [17]. Actually it does not measure an error but a translation accuracy, so its opposite is to be minimised.

3.3. Downhill simplex implementation details

We implemented the simplex method according to [5]. The method uses a geometrical figure called a $simplex$ consisting, in N dimensions, of $N+1$ points and all their interconnecting line segments, polygonal faces, etc. The starting point is a set of $N+1$ points in parameter space, defining an initial simplex. At each step, the simplex performs geometrical operations (reflexions, contractions and expansions) until a local minimum is reached.

Given a starting point $\mathbf{P}_0$ (see section 3.2), the other N points of the initial simplex were taken to be $\mathbf{P}_i = \mathbf{P}_0 + \alpha_i \mathbf{e}_i$, where the $\mathbf{e}_i$ are unit vectors. The N constants α_i were chosen randomly such that the perturbed parameter $P_{0i} + \alpha_i$ be in the range corresponding to this scaling factor (as defined in section 3.2). For each of the seven starting points $\mathbf{P}_0$, we ran the algorithm from 10 different initial simplexes. Different initial simplexes were obtained by varying the seed of the random generator used to compute the α_i constants.

3.4. SPSA implementation details

After some experiments, we adopted slight changes to the form of the algorithm presented in section 2. The algorithm presented in section 2 does not restrict the updated set of parameters λ at a new iteration. This means that if we are unlucky with the $\mathbf{\Delta_k}$ vector, we can go back from an good λ vector to a bad λ vector. This process will eventually converge, but it can take many iterations. Thus we introduced a function evaluation after step 5, to be able to determine whether the new parameters led to an improvement or not. A new set of parameters λ_{k+1} which was worse than the current one (λ_k) was accepted according to the following probability distribution:

$$\exp\left[-\frac{|E(\hat{\lambda}_{k+1}) - E(\hat{\lambda}_k)|}{T\,(k+1)}\right], \qquad (3)$$

were T was empirically set to 0.005. According to this probability distribution, the worse the new set of lambdas, the less probability to accept it.

Since we introduced an evaluation of $E(\hat{\lambda}_k)$, we replaced the two-sided gradient approximation by a one-sided one, which involves evaluations of $E(\hat{\lambda}_k)$ and $E(\hat{\lambda}_k + \text{perturbation})$. If the noise caused by the one-sided approximation (as opposed to the two-sided approximation) is small compared to the noise arising from the simultaneous approximation, one-sided approximation can be more efficient since it saves one function evaluation at each iteration.

Finally, after a certain number of iterations without improving the optimum, we also changed the distribution in Step 2 to a $0, \pm 1$ distribution with probability $1/3$ for each outcome. Although this distribution seems to slow down the algorithm, it allows for a finer approximation of the gradient in a subpart of the parameter space.

At the end the SPSA algorithm was implemented as follows:

Step 1 Calculate gain sequences a_k and c_k. Notice that the choice of the gain sequence a_k and c_k are critical to the performance of SPSA. We choose the basic parameters according to [18], and tuning the algorithm over various development sets (distinct from the one used in this experiment) from different machine translation tasks. Thus these parameters are expected to be valid for experiments with the same objective function on other language pairs and corpora. We used $a_k = 8/(2+k+1)^{0.602}$ and $c_k = 0.25/(k+1)^{0.101}$.

Step 2 Generate the simultaneous perturbation vector $\mathbf{\Delta}_k$. Each component of $\mathbf{\Delta}_k$ was a Bernoulli ± 1 distribution with probability of $1/2$ for each ± 1 outcome (or a $0, \pm 1$ distribution with probability $1/3$ for each outcome, as mentioned above).

Step 3 Evaluate $E(\hat{\lambda}_k + c_k \mathbf{\Delta_k})$

Step 4 Approximate the gradient as in equation 2, but replacing $E(\hat{\lambda}_k - c_k \mathbf{\Delta_k})$ by $E(\hat{\lambda}_k)$ and dividing by c_k instead of $2c_k$.

Step 5 Update λ estimate as in equation 1, and evaluate the objective function with this new set of parameters. Accept the new parameters according to the probability distribution in equation 3.

Step 6 Iteration or termination (as in section 2).

In our comparative experiment, for each starting point $\mathbf{P}_0$, we ran the algorithm with 10 different seeds for the random generator which computes the simultaneous perturbation vector in step 2. These seeds were the same as those used to generate the 10 initial simplexes (see subsection 3.3).

3.5. Data set

The translation system was trained with the Chinese-English data provided for IWSLT'06 evaluation campaign, and the parameters were tuned over the development set provided for the same evaluation (dev4). These parameters were then used to translate the test set, which was a selection of 500 sentences among the development sets of previous evaluations (dev1, dev2 and dev3). Table 2 shows the main statistics of the training, development and test data used, including number of references, number of sentences, number of words, vocabulary and average sentence length for each language.

	sent	words	vocab.	avg len
Training set				
Chinese	46k M	314k	9725	6.7
English		326k	9643	7.0
Development set (7 references)				
Chinese	489	5478	1096	11.2
Test set (16 references)				
Chinese	500	3005	909	6.0

Table 2: *Training, development and test sets statistics.*

4. Results

We report in table 3 the average BLEU score and standard deviation obtained after running 20, 40, 60 and 90 function evaluations of the simplex algorithms and SPSA algorithm. When an optimisation converged before the given number of function evaluations, the optimum value was taken. In each cell of table 3, the upper number refers to the simplex, and the lower refers to SPSA. For each initial set of parameters, average and standard deviation are calculated over the 10 slightly different realisations controlled with the random seeds.

First, from table 3 it seems that both algorithms have similar performance in terms of the optimum value achieved after a given number of function evaluations. Nevertheless, it is remarkable that from 60 function evaluations on, the standard deviation is always smaller for the SPSA algorithm, which suggests that this is a more stable method. Since the implementation of the different realisations cannot be the same for both algorithms, we cannot be totally certain that it was fair. A change of seed to generate the simultaneous perturbation for the SPSA may be less significant than a change of initial simplex. To verify this, we need to fix the seed and see how the algorithm behaves across several initial points. A first indication is given by the last row of table 3. In this row, the average and standard deviation of the averages $< BLEU >_{Ni}$ taken after N function evaluations, for each point i, are calculated. The standard deviation of the averages, after 60 function evaluations, is much lower with SPSA method, which confirms that on this data set, the simplex optimal value is more dependent on the starting parameters than SPSA.

Table 4 explores this point in greater detail. For a given

Pt	Function Evaluations			
	20	40	60	90
1	17.9±0.45	18.4±0.55	18.7±0.47	18.9±0.47
	17.6±0.74	18.9±0.53	19.4±0.22	19.7±0.15
2	18.8±0.40	19.1±0.41	19.2±0.41	19.3±0.42
	18.9±0.48	19.3±0.15	19.5±0.11	19.6±0.16
3	19.1±0.22	19.3±0.26	19.5±0.29	19.5±0.30
	19.1±0.44	19.4±0.21	19.5±0.19	19.6±0.10
4	18.9±0.45	19.4±0.39	19.6±0.34	19.7±0.34
	18.7±0.63	19.5±0.28	19.6±0.18	19.7±0.11
5	19.4±0.23	19.6±0.23	19.6±0.24	19.6±0.25
	19.4±0.19	19.5±0.17	19.5±0.17	19.6±0.14
6	19.5±0.19	19.7±0.20	19.7±0.20	19.8±0.20
	19.3±0.20	19.5±0.20	19.6±0.16	19.7±0.10
7	19.6±0.13	19.7±0.15	19.8±0.14	19.8±0.15
	19.4±0.19	19.6±0.12	19.7±0.11	19.7±0.11
	19.0±0.58	19.3±0.45	19.4±0.37	19.5±0.33
	18.9±0.64	19.4±0.22	19.5±0.09	19.7±0.08

Table 3: Average BLEU score and standard deviation obtained with the simplex method (above) and SPSA method (below) in the development set, after 20, 40, 60, and 90 function evaluations, for each initial point in parameter space (referred to as pt). In the last row, separated from the rest of the table, the average and standard deviation of the averages are displayed.

seed used, determining a given realisation, the only varying factor is the starting point. For each seed, the average BLEU score and standard deviation over all 7 starting points, after 20, 40, 60, and 90 function evaluations, are shown. For all seeds, after at least 60 function evaluations, the optimum value obtained with SPSA is less sensitive to the choice of initial parameters, which should lead to more consistent results. For SPSA, the highest standard deviation after 90 function evaluations is 0.18. For the simplex, it reaches 0.67. Thus doing two successive optimisations, one can expect in average up to 0.4 percent BLEU difference with SPSA and up to more than 1.3 percent BLEU different with the simplex. The conjugated effect of two SPSA properties not shared by the simplex method may contribute to explain this difference in stability. Firstly, SPSA search path follows in average the direction of the gradient, whereas the simplex orientation is blind. Secondly, SPSA has always a probability to go away from a zone close to a minimum, which allows it to find a lower minimum elsewhere in the search space. On the contrary, when the simplex shrinks in a zone close to a minimum, it is stuck in that zone.

While optimal objective function values lie in a pretty close range, as seen in Table 3, Figure 1 show that final values in parameter space are very dispersed. Thus different parameter sets lead to similar scores. Surprisingly, the value of the lexical (L1) model weight does not seem to be determi-

ID	Function Evaluations			
	20	40	60	90
1	19.0±0.76	19.5±0.22	19.6±0.20	19.7±0.22
	18.9±1.09	19.5±0.31	19.7±0.14	19.7±0.06
2	18.8±0.65	19.1±0.69	19.1±0.68	19.1±0.67
	19.0±0.68	19.5±0.26	19.6±0.16	19.6±0.14
3	19.1±0.44	19.2±0.36	19.4±0.30	19.5±0.28
	19.1±0.40	19.4±0.23	19.6±0.25	19.6±0.18
4	18.9±0.88	19.3±0.73	19.6±0.43	19.7±0.26
	18.8±0.60	19.3±0.29	19.5±0.16	19.6±0.14
5	19.1±0.61	19.3±0.57	19.5±0.41	19.6±0.38
	18.5±1.11	19.2±0.74	19.5±0.23	19.7±0.11
6	19.1±0.38	19.4±0.40	19.5±0.40	19.6±0.38
	18.9±0.98	19.4±0.18	19.6±0.19	19.7±0.18
7	18.8±0.73	19.1±0.66	19.2±0.64	19.3±0.60
	19.1±0.42	19.3±0.33	19.5±0.18	19.6±0.14
8	19.1±0.47	19.4±0.36	19.5±0.40	19.6±0.36
	18.9±0.89	19.4±0.26	19.5±0.16	19.7±0.14
9	19.0±0.72	19.3±0.68	19.5±0.58	19.5±0.57
	18.9±0.54	19.5±0.19	19.6±0.17	19.7±0.11
10	19.1±0.71	19.4±0.59	19.5±0.36	19.6±0.32
	18.9±0.65	19.4±0.28	19.5±0.16	19.6±0.16

Table 4: Average BLEU score and standard deviation obtained with the simplex method (above) and SPSA method (below) in the development set, after 20, 40, 60, and 90 function evaluations, for each seed used to generate different algorithm conditions. Only seed ID numbers are displayed.

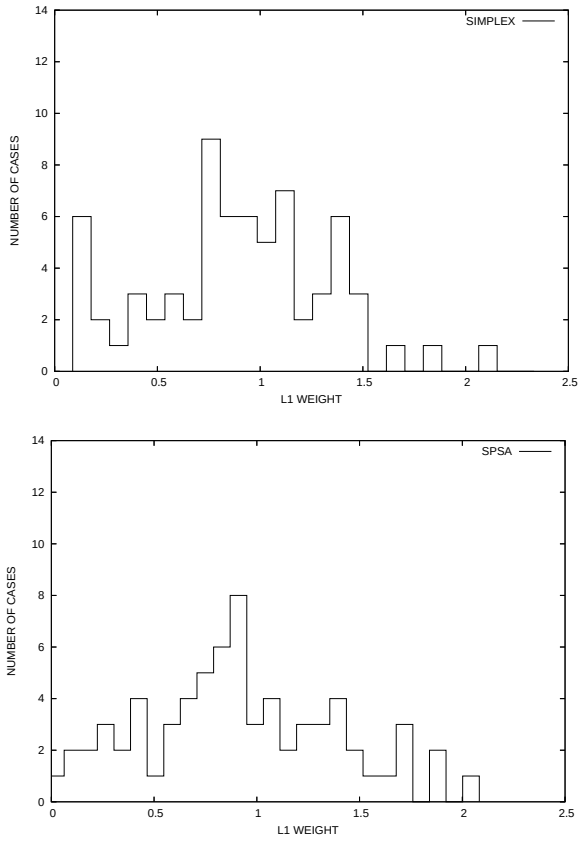


Figure 1: Histogram of L1 model weights for simplex (above) and SPSA (below)

nant, although this model has got a big impact in translation quality [19]. This is an indication of the interdependence of the various models. Figure 1 also suggests that there would be no point in averaging parameter values in order to gain generalisation power.

As ultimate goal, we need to see if the stability of SPSA optimisations is conserved when translating new text (*i.e.* the test corpus). For each initial point and seed, and after a given number of function evaluations, we collected the optimum parameter set over the development corpus, and translated the test corpus with these parameters. Results are brought together in Table 5. Table 5 instructs, as expected, that dispersion of scores is higher in test than in development. It also reveals that the standard deviation in test is similar for both algorithms. Thus the stability gain observed in the development corpus for SPSA was not conserved with new data. Finally, the last row of Table 5 indicates that the average BLEU score in test is similar for both algorithms.

Since we have got 10 realisations of the algorithms for each initial set of parameters, it is interesting to select the parameters corresponding to the best score out of the 10 realisations, and translate the test corpus with these parameters. The results are plotted in Figure 2. Firstly, we can notice that the selected parameters do not lead to substantially better results than the average of all results (reported in the last row of Table 5). The largest gain is 0.2 point BLEU for SPSA after 60 function evaluations. Secondly, while no over-fitting seems to occur, continuing the optimisation over development data after 30 function evaluations does not lead to significant translation quality improvement in test. Even 10 function evaluations were sufficient to obtain nearly optimal parameters with both algorithms.

5. Conclusions and further work

We have presented experiments in which the SPSA algorithm has been used to tune SMT parameters. These experiments have been repeated with the downhill simplex method for comparison. According to the results obtained in this task, both methods seem to have similar performance. However, SPSA was more robust than the simplex with respect to the choice of initial parameters and with respect to slightly different realisations of the algorithm. This conclusion is not restricted to a particular tuning setup. However, this SPSA advantage was not conserved when using the optimal parameters to translate new data. We also observed a high dispersion in parameter space, showing that various sets of param-

Pt	Function Evaluations			
	20	40	60	90
1	41.1±1.08	41.4±1.10	41.4±1.03	41.4±1.09
	40.3±1.12	41.2±0.61	41.6±0.37	41.7±0.52
2	42.3±0.56	42.4±0.65	42.5±0.68	42.5±0.70
	42.1±0.84	42.3±0.78	42.3±0.91	42.3±0.87
3	41.8±0.55	42.0±0.60	42.0±0.60	42.0±0.65
	41.9±0.77	41.9±0.66	42.0±0.69	41.9±0.70
4	41.5±0.55	41.8±0.27	41.8±0.35	41.8±0.34
	41.9±0.65	42.2±0.34	42.1±0.37	42.0±0.38
5	42.5±0.37	42.3±0.46	42.3±0.57	42.3±0.51
	42.2±0.47	42.1±0.60	41.8±0.50	42.0±0.44
6	41.8±0.55	41.9±0.33	42.1±0.29	42.1±0.35
	42.0±0.56	42.1±0.63	42.0±0.56	42.0±0.47
7	42.4±0.32	42.5±0.34	42.5±0.33	42.5±0.31
	42.2±0.50	42.2±0.35	42.1±0.30	42.3±0.33
	41.9±0.52	42.0±0.40	42.1±0.39	42.1±0.38
	41.8±0.69	42.0±0.38	42.0±0.25	42.0±0.22

Table 5: Average BLEU score and standard deviation obtained with the simplex method (above) and SPSA method (below) in the TEST set, after 20, 40, 60, and 90 function evaluations, for the parameters obtained in development for each initial point (referred to as pt). In the last row, separated from the rest of the table, the average and standard deviation of the averages are displayed.

eters led to similar scores.

While no over-fitting was noticed, nearly optimum results in test could be obtained after only 10 function evaluations over the development corpus. The dispersion of results in test may be overvalued because of the task, which allows particularly poor generalisation since training, development and test corpora are small. Thus, it would be interesting to repeat these experiment with more data, such as those of European Parliament corpus. Furthermore, SPSA being expected to perform better for a problem of higher dimensionality, we should carry out experiments with a system including more feature functions. Finally, we are planing to perform a sensitiveness analysis with neural networks, in order to study the impact in translation score (of the development corpus) resulting from perturbations of each parameter.

6. Acknowledgements

This work has been partially funded by the European Union under the integrated project TC-STAR - Technology and Corpora for Speech to Speech Translation -(IST-2002-FP6-506738, http://www.tc-star.org). The authors also want to thank Dr. Héctor Klie, from the Center for Subsurface Modeling of The University of Texas at Austin, for suggesting the use of SPSA in the context of Statistical Machine Translation.

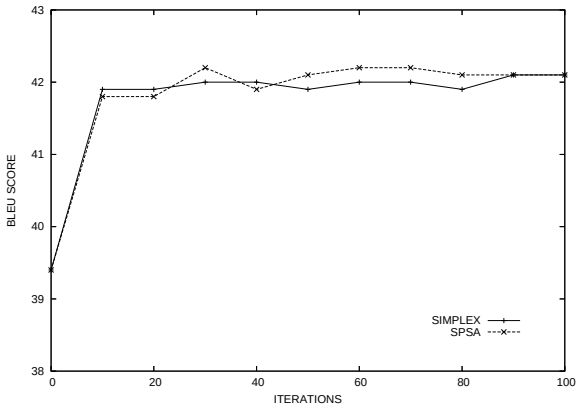


Figure 2: Average, over the initial points, of the results in test for the best parameters obtained in development among the 10 different realisations.

7. References

[1] P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer, "The mathematics of statistical machine translation: Parameter estimation," *Computational Linguistics*, vol. 19, no. 2, pp. 263–311, 1993.

[2] F. Och and H. Ney, "Dicriminative training and maximum entropy models for statistical machine translation," in *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA, July 2002, pp. 295–302.

[3] F. Och, "Minimum error rate training in statistical machine translation," in *Proc. of the 41th Annual Meeting of the Association for Computational Linguistics*, 2003, pp. 160–167.

[4] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, pp. 155–162, 1964.

[5] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C++: the Art of Scientific Computing*. Cambridge University Press, 2002.

[6] J. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, pp. 308–313, 1965.

[7] M. Cettolo, M. Federico, N. Bertoldi, R. Cattoni, and B. Chen, "A look inside the itc-irst smt system," in *Proc. of Machine Translation Summit X*, Phuket, Thailand, September 2005, pp. 451–457.

[8] N. Bertoldi, "Minimum error training (updates)," Slides of the JHU Summer Workshop (http://www.statmt.org/jhuws), 2006.

[9] K. Kirchhoff and M. Yang, “Improved language modeling for statistical machine translation,” in *Proc. of the ACL Workshop on Building and Using Parallel Texts*, Ann Arbor, Michigan, June 2005, pp. 125–128.

[10] J. Mariño, R. Banchs, J. M. Crego, A. de Gispert, P. Lambert, J. Fonollosa, and M. Ruiz, “Bilingual n-gram statistical machine translation,” in *Proc. of Machine Translation Summit X*, Phuket, Thailand, 2005, pp. 275–82.

[11] E. Matusov, R. Zens, D. Vilar, A. Mauser, M. Popovic, and H. Ney, “The rwth machine translation system,” in *Proc. of the TC-STAR Workshop on Speech-to-Speech Translation*, Barcelona, Spain, June 2006, pp. 31–36.

[12] L. Han and M. Neumann, “Effect of dimensionality on the Nelder-Mead simplex method,” *Optimization Methods and Software*, vol. 21, no. 1, pp. 1–16, 2006.

[13] J. M. Crego, A. de Gispert, and J. Mariño, “The TALP ngram-based SMT system for IWSLT’05,” Pittsburgh, USA, October 2005, pp. 191–198.

[14] J. C. Spall, “Multivariate stochastic approximation using a simultaneous perturbation gradient approximation,” *IEEE Trans. Automat. Control*, vol. 37, pp. 332–341, 1992.

[15] ——, “An overview of the simultaneous perturbation method for efficient optimization,” *Johns Hopkins APL Technical Digest*, vol. 19, no. 4, pp. 482–492, 1998.

[16] J. M. Crego, A. de Gispert, P. Lambert, M. R. Costa-jussà, M. Khalilov, R. Banchs, J. B. Mariño, and J. A. R. Fonollosa, “N-gram-based smt system enhanced with reordering patterns,” in *Proc. of the HLT-NAACL Workshop on Statistical Machine Translation*. New York City: Association for Computational Linguistics, June 2006, pp. 162–165.

[17] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” IBM Research Report, RC22176, 2001.

[18] J. C. Spall, “Implementation of the simultaneous perturbation algorithm of stochastic optimization,” *IEEE Trans. Aerospace and Electronic Systems*, vol. 34, no. 3, pp. 817–823, 1998.

[19] J. B. Mariño, R. Banchs, J. M. Crego, A. de Gispert, P. Lambert, J. A. R. Fonollosa, and M. R. Costa-jussà, “N-gram based machine translation,” *Computational Linguistics*, vol. 32, no. 4, 2006.