# Robust Estimation of Feature Weights in Statistical Machine Translation

**Cristina España-Bonet and Lluís Màrquez**
TALP Research Center
Universitat Politècnica de Catalunya
Jordi Girona 1–3, 08034 Barcelona
cristinae@lsi.upc.edu, lluism@lsi.upc.edu

## Abstract

Weights of the various components in a standard Statistical Machine Translation model are usually estimated via Minimum Error Rate Training. With this, one finds their optimum value on a development set with the expectation that these optimal weights generalise well to other test sets. However, this is not always the case when domains differ. This work uses a perceptron algorithm to learn more robust weights to be used on out-of-domain corpora without the need for specialised data. For an Arabic-to-English translation system, the generalisation of weights represents an improvement of more than 2 points of BLEU with respect to the MERT baseline using the same information.

## 1 Introduction

In Statistical Machine Translation (SMT) and within the log-linear model (Och and Ney, 2002), the best translation $\hat{e}$ for a given source sentence $f$ is the most probable one, and the probability is expressed as a weighted sum of different elements:

$$T(f) = \hat{e} = \mathrm{argmax_e} \sum_m \lambda_m h_m(f, e) \,. \qquad (1)$$

In the standard most simple form, one considers 8 components being $h_m(f, e)$ log-probabilities of: the language model $P(e)$, the generative and discriminative lexical translation probabilities $lex(f|e)$ and $lex(e|f)$ respectively, the generative and discriminative translation models $P(f|e)$ and $P(e|f)$, the distortion model $P_d(e, f)$, and the phrase and word penalties, $ph(e)$ and $w(e)$.



The $\lambda$ weights, which account for the relative importance of each feature in the log-linear probabilistic model, are commonly estimated by optimising the translation performance on a development set. For this optimisation one can use Minimum Error Rate Training (MERT) (Och, 2003) where BLEU (Papineni et al., 2002) is the reference score.

MERT estimates the 8D best fit by searching the minimum in each dimension of the parameter space. The line search used in Och (2003) is demonstrated to find the absolute minimum in that direction, still, this does not guarantee that the best parameters obtained are the optimum ones. In fact, the larger the number of features, the less reliable the global minimisation will be. Some works such as Cer et al. (2008), Moore and Quirk (2008), or Foster and Kuhn (2009) try to improve the standard MERT minimisation. Here we do not follow this line, since we are not interested in finding the optimal parameters on development but on test.

In this study, we see how parameters estimated with MERT can generalise quite bad on test sets that depart substantially from the training and development sets. Our goal is to find a more robust vector of weights $\overrightarrow{\lambda}$ that, even without being optimal on development or test when the domain is akin, better generalise on the other cases. We show empirically that this can be done by including machine learning techiques to estimate $\overrightarrow{\lambda}$ without the need for data in the new domain. However, if one has at his disposal such data sets, systems can be easier adapted. As there exist domain adaptation methods to improve results when these data sets are available, we devote most of the work to the first case, but we also check that the method does not hurt the performance in the second case.

For this purpose, we complement the standard minimisation methods with an averaged perceptron-based re-estimation of parameters. Perceptrons have been used before with the aim of adding a large amount of new features to statistical systems avoiding the problem of the numerical minimisation of such a large vector of parameters (Liang et al., 2006; Tillmann and Zhang, 2006; Arun and Koehn, 2007). Other algorithms such as MIRA have been used for the same purpose (Arun and Koehn, 2007; Chiang et al., 2008). Here, the philosophy is different. We do not intend to include new information, but to profit better the available data as we will argue in the following. Even using the same data sets, the combination of MERT and the perceptron training can improve more than 2 points of BLEU the result of MERT alone. When including specialised data for development, the difference between MERT and the combined training is not so spectacular, but, still, the perceptron stage attains the leading results.

The outline of the paper is as follows. Section 2 introduces the perceptron training, details the algorithm and shows different forms for the update rule and the choice of a gold standard, two key aspects of using this algorithm for machine translation. Section 3 describes and classifies the data used in the analysis. Afterwards, in Section 4, we detail our experiments. The first one, Cross-domain testing, is devoted to demonstrate how one can enhance his system for an out-of-domain test set by appending a perceptron training. The second experiment focuses on using an out-of-domain development set for tuning the system into the new domain. Finally, we draw our conclusions in Section 5.

## 2 Perceptron-based training

The averaged structured perceptron (Collins, 2002) is an online mistake driven algorithm that determines the weights of a linear feature function by correcting their values according to the distance to the true solution.

The score function that quantifies the quality of a translation in SMT, Eq. 1, is a linear function of the $h_m$ components. Therefore, the corresponding weights can be learned with the perceptron.

Figure 1 details the perceptron algorithm. Given the training data set $\{f^i,e^i\}$, an initial value for the weights $\overrightarrow{\lambda}_0$, the learning rate $\epsilon$, and the number of epochs $N$, the perceptron translates (decodes) every sentence in the training set and compares the obtained translation with the target. The function `decode()` calculates the score and the corresponding argmax in Eq. 1 using the current weights. It returns an $n$-best list of translations ê ranked by its score. The guess is the first element in the list. The target is for us the element with the highest BLEU in the list. For each example during the training of the perceptron, weights are updated whenever the features of the guess $\overrightarrow{h}(f_i, \text{guess})$ are different from the features of the target $\overrightarrow{h}(f_i, \text{tgt})$. The update is a function of the difference between the corresponding features, $\Delta\overrightarrow{h}$. In the end, the algorithm returns the averaged weights $\overrightarrow{\Lambda}$.

The next words discuss some possible update rules and choices of the target or gold standard for the machine translation problem.

**Input:**
  Training data, $\{f^i,e^i\}_{i=1}^{T}$
  Initial weights, $\overrightarrow{\lambda}_0$
  $N$ epochs, learning rate $\epsilon$

**for each epoch** $n = 1, ..., N$
  **for each example** $f_i$ $i = 1, ..., T$
    ê = decode($f_i$,$\lambda_i$)
    guess: ê[1]
    tgt: $\text{argmax}_j\,(\text{BLEU}(\hat{\mathbf{e}}[j]))$
    **if** $\overrightarrow{h}(f_i, \text{guess}) \neq \overrightarrow{h}(f_i, \text{tgt})$ **then**
      $\overrightarrow{\lambda}_i := \overrightarrow{\lambda}_i + \epsilon \cdot \Delta\overrightarrow{h}(f_i, \text{tgt}, \text{guess})$
    **end if**
    $\overrightarrow{\Lambda} := \overrightarrow{\Lambda} + \overrightarrow{\lambda}_i$
  **end for**
**end for**
**return** $(\overrightarrow{\Lambda}/NT)$

Figure 1: Average perceptron algorithm applied to the weights $\lambda$.

### 2.1 Update rule

The traditional update rule for the perceptron accounts for the difference between the features of the guess and the target modulated by a learning rate $\epsilon$. We train the perceptron with the eight real features related to the probabilities appearing in Equation 1. That is, $h_m$ = [$P(e)$, $P(e|f)$, $P(f|e)$, $lex(e|f)$, $lex(f|e)$, $P_d(e,f)$, $ph(e)$, $w(e)$]. The variation among the same feature of different translations can be huge, and that can easily cause the value of the corresponding weight to shoot up unless it is smoothed by the learning rate. These huge discrepancies are characteristic of the SMT problem and we consider three alternatives to address the issue.

1. The standard update rule. Every weight is treated in a same way using a unique learning rate. The amount of change is just the difference between features.

$$\overrightarrow{\lambda}_i := \overrightarrow{\lambda}_i + \epsilon \cdot \left[\overrightarrow{h}(f_i, \text{tgt}) - \overrightarrow{h}(f_i, \text{guess})\right]$$

2. The normalised update rule. The difference between features is normalised to the sum of that feature for all the previous examples. That makes an effective learning rate that depends on the feature.

$$\overrightarrow{\lambda}_i := \overrightarrow{\lambda}_i + \frac{\epsilon'}{\frac{\sum_{k=1}^{i-1} |\overrightarrow{h}_{(f_k, \text{tgt})} - \overrightarrow{h}_{(f_k, \text{guess})}|}{i-1}} \cdot \left[\overrightarrow{h}(f_i, \text{tgt}) - \overrightarrow{h}(f_i, \text{guess})\right]$$

We checked that, at the end, the steep changes produced in number 1 and number 2 can be compensated by a low learning rate, being the effective update of the same order of magnitude. The same happens for the third update rule we consider, which is the one we use for our final experiments due to its simplicity:

3. The constant update rule. This choice only makes use of the fact that two features are different and the direction of change without considering how much. The amount for updating the weights is the same for all the features and the examples, and it is equal to the learning rate.

$$\overrightarrow{\lambda}_i := \overrightarrow{\lambda}_i + \epsilon \cdot \text{sign}\Big(\overrightarrow{h}(f_i, \text{tgt}) - \overrightarrow{h}(f_i, \text{guess})\Big)$$

Other variations of the traditional update rule for the perceptron can be found in Tillmann and Zhang (2006), but they have not been included in our analysis.

### 2.2 Gold Standard or target

The job of the update rule is to bring closer the guess sentence towards the correct solution, the gold standard or target, but in machine translation this is not always possible. The reference translation one should optimise towards could be simply not reachable because not all the necessary phrases to construct the sentence are present in the translation table. Besides, the opposite phenomena is also common: a same translation is reachable through multiple phrase combinations. In the former case one lacks the corresponding $h_m$; in the latter one has multiple $h_m$'s for the same sentence.

Several approaches have been used to address this problem – see for instance Liang et al. (2006), Tillmann and Zhang (2006), Arun and Koehn (2007). A common solution is to use as gold standard an approximation of the best reachable translation. This one is simulated by using the best BLEU sentence in the $n$-best translation list. Notice that one must use a smoothed BLEU score at the sentence level in this case, whereas the MER training evaluates the translations at the document level. So, for our principal choice:

1. The gold standard is the sentence with the highest BLEU score in the $n$-best list (local update in Liang et al. (2006)).

But other alternatives are possible and have been tested as well:

2. All the sentences with a higher BLEU score than the guess are considered, and the weights are updated for all these solutions.

3. As in 2., one considers all the sentences with a higher BLEU than the guess, but there is at most one update. This is done whenever a percentage of the sentences with higher BLEU than the guess (60%, 80% and 90% in our experiments) agrees with the direction of the update.

Since in our experiments the first option clearly overcame the other two, Section 4 reports the results for this more convenient choice.

## 3 Data

The system is trained and evaluated for the Arabic-to-English translation task, a language pair with multiple development and test sets available. The training set is a compilation of six newswire corpora supplied by the Linguistic Data Consortium for the *2008 NIST Machine Translation Open Evaluation*[1]. We select 124k lines with a length shorter than 100 words. The language model is estimated from the English side of the corpus with `SRILM` (Stolcke, 2002) and the alignment is done

[1] `http://www.nist.gov/speech/tests/mt/2008`

| Test set | Segs. | Genre | OOVs (%) | Perplexity Ara/Eng |
|---|---|---|---|---|
| Trdev | 500 | N | 1.25 | 272/129 |
| Trtest | 500 | N | 1.18 | 270/133 |
| N05 | 1056 | N | 2.02 | 320/145 |
| N06 | 1797 | NW | 5.16 | 598/205 |
| N08 | 1357 | NW | 3.82 | 568/227 |

Table 1: Test sets used in the analysis and their perplexity given the newswire training set. Sets are classified according to their genre (N stands for Newswire and NW for News+Web).

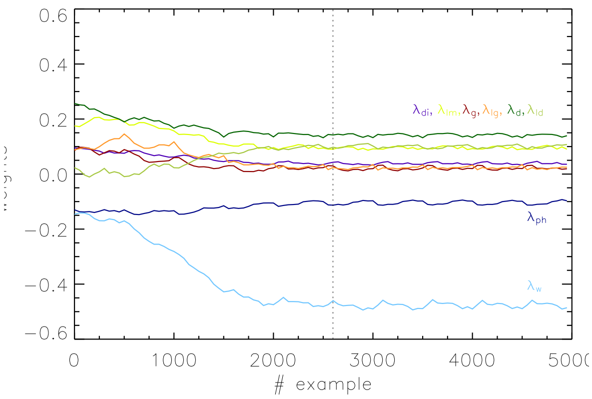


Figure 2: Evolution of the eight weights during the perceptron training.

with GIZA++ (Och and Ney, 2003). The phrase extraction and decoding are done with the Moses package (Koehn et al., 2006; Koehn et al., 2007).

For development and test we use several sets as shown in Table 1. The similarity of each of the sets to training set is quantified by the perplexity with its language model. *Trdev* and *Trtest* are drawn from the same compilation as the training set, and therefore show the lowest perplexity. The test set for the 2005 NIST MT Evaluation (*N05*) shares the genre with the training set and is relatively similar in perplexity. From now on, we say that a data set is similar to the training corpus whenever the perplexity is low. On the other hand, both *N06* and *N08* include segments from the web, the number of out of vocabulary words (OOVs) augments, and they show larger differences with the training. The goal of our method is to optimise the weights so that they obtain the optimum results on these sets.

## 4 Experiments

We test our approach on the test sets introduced in the previous section. As a common setting, our experiments use a learning rate of 0.001 and consider an $n$-best list with 100 candidate translations. As it has been argued in Sections 2.1 and 2.2, we report the results obtained with update rule #3 and choice of gold standard #1. These choices show a good time-quality trade-off[2].

### 4.1 Cross-domain testing

We focus on the case where we only have data of a given domain but want to translate a test set of a different genre. Contrary to domain adaptation techniques, this is a complete empirical approach that does not make use of any out-of-domain data. So, we are constrained to train our system with the training set described in Section 3 and to use *Trdev* for development.

The execution of MERT on Trdev provides us with a vector of weights $\overrightarrow{\lambda}$ appropriate to use on Trtest, but we look for a more robust vector that can be applied on different test sets. Therefore, we do a second stage of development, the perceptron training, which takes as initial point the result of MERT.

During the perceptron training, the BLEU score diminishes from the maximum value given by MERT. The perceptron is not able to find better parameters for the development set, but weights are being perturbed from its original value example by example. These variations during the training can be seen in Figure 2. The plot shows the evolution along 5,000 examples which correspond to ten iterations of the perceptron. One can observe a smooth pattern repeated through iterations, but the randomisation of the order of the sentences in Trdev does not alter the global trend of the evolution and the effects on test are minimum.

From around example number 2,000 on, weights show a stable behaviour. We consider the weights to be stable and stop the training when the variation of all of them is less than 0.00005, 0.05% of its variation along examples, which, with our constant update rule, is equivalent to 0.05% of the learning rate. In this case, the stopping criterion lets the perceptron run up to example 2,598 (vertical dotted line in Fig. 2), a bit more than 5 iterations.

Other stopping criteria can be used. For instance, Arun and Koehn (2007) run 10 iterations

[2]Current implementation of the perceptron is external to the decoder, and the execution time is dominated by decoding time.

| | in-domain | | out-of-domain | |
|---|---|---|---|---|
| | Trtest | N05 | N06 | N08 |
| M | **23.87** | 43.76 | 30.24 | 29.06 |
| M+P | 22.77 | **44.06** | **32.08** | **31.52** |
| M on test | 24.27 | 45.46 | 32.96 | 32.77 |

Table 2: BLEU scores for several tests as obtained by MERT (first row) and by the combined training with the perceptron (second row). The lower row shows the result of MERT on the test set.

of the perceptron that are evaluated on a development set. The best value there is used for testing. We cannot use this criterion because our perceptron is initialised with MERT results, so, whenever we use the same set for MERT and the perceptron, the BLEU evolution will usually be negative since MERT has already found a good value for the weights on that set if not the optimum. As an example, for the current training, the BLEU attained by MERT is 26.20 whereas when the stopping criterion is met the perceptron reaches 24.63. In these cases and according to this criterion, we should stop iterating at the beginning instead and there would be no generalisation, whereas our results show that a decrement in the BLEU score on the development set can still correspond to an increment on test.

Using the weights at this point, we estimate the BLEU score for four test sets. Table 2 gathers these results and shows them together with those obtained by MERT alone. The last row of the table shows the artificial result of MERT on the test set. We consider this value to be an upper limit to the attainable BLEU score for the given test set.

The first thing to notice in Table 2 is that for test sets similar to the training corpus there is no benefit in continuing with the perceptron training. That is the case of Trtest. There is little room for improvement because the upper bound that hints MERT on Trtest is close to that obtained with the development weights and, besides, we are not interested in generalising the value of the weights when they belong to the same domain. This is why the BLEU score gets no benefit with the combined training. A similar thing happens with N05, the domain is akin but now the upper bound is higher, and, because of this, there is a slight improvement which is already noticeable although not large.

On the other hand, data sets that add segments from the web, N06 and N08, do profit from the perceptron training. Note that we are reusing the same development set as for MERT, and there is around two BLEU points of improvement with respect to the result with MERT alone, 1.84 points for N06 and 2.46 for N08. Since we do not use any data similar to the test sets for development, the BLEU scores are still one point below the upper limit that the straightforward application of MERT on the test sets would give. This confirms that moving slightly away from the optimal value on development but keeping the information of the data set is beneficial for generalising, and this is accomplished by the perceptron. An early stopping of MERT does not have the same effect. We checked that the almost monotonous increment of BLEU throught MERT iterations on development sometimes translates into erratic BLEU results on test, especially when the domain of the data sets differs. The variance of BLEU scores on test can be large. There are values quite better than the last one but also quite worse, and there is no way to know when to obtain the best one from the training.

In order to find out whether the results are statistically significant, we generated 1,000 sets by pair bootstrap resampling of the original test sets (Koehn, 2004). All the enhancements with respect to the MERT baseline result to be significant and are written in boldface in Table 2.

Since the perceptron is maximising the BLEU score, it is on this metric that we mostly analyse the results, but the quality of the translation cannot be only judged in terms of BLEU. We thus investigate if the positive effects are also captured by other metrics. Table 3 summarises the results for a set of lexical metrics: WER (Nießen et al., 2000), BLEU (Papineni et al., 2002), NIST (Doddington, 2002), ROUGE (Lin and Och, 2004) and METEOR (Banerjee and Lavie, 2005). The last metric, ULC (Giménez and Amigó, 2006), performs a linear combination of a set of 33 lexical metrics, most of them variants of the ones appearing in the table (see Giménez (2007) for details).

As a general trend, the same conclusions seen with BLEU can be extracted here. For the out-of-domain test sets, the addition of the perceptron stage improves for all the metrics but WER the results with respect to MERT alone. For in-domain test sets, the answer is not unique and the behaviour depends on the metric, so, there is no a clear effect of the second stage as the similarity of the ULC scores suggests.

| | Trtest | | N05 | | N06 | | N08 | |
|---|---|---|---|---|---|---|---|---|
| Metric | M | M+P | M | M+P | M | M+P | M | M+P |
| 1-WER | 0.3124 | 0.2802 | 0.5230 | 0.5043 | 0.4359 | 0.4239 | 0.4174 | 0.4038 |
| BLEU | 0.2387 | 0.2277 | 0.4376 | 0.4406 | 0.3024 | 0.3208 | 0.2906 | 0.3152 |
| NIST | 6.3772 | 6.1505 | 10.375 | 10.286 | 8.4585 | 8.9264 | 8.5186 | 9.0402 |
| ROUGE | 0.3076 | 0.3093 | 0.3027 | 0.3048 | 0.2720 | 0.2733 | 0.2642 | 0.2681 |
| METEOR | 0.5119 | 0.5226 | 0.6353 | 0.6493 | 0.5312 | 0.5489 | 0.5310 | 0.5552 |
| ULC | 0.7235 | 0.7100 | 1.1320 | 1.1297 | 0.8522 | 0.8876 | 0.9287 | 0.9723 |

Table 3: Automatic evaluation of the M and M+P systems on four test sets using different lexical metrics. M represents the MERT baseline with Trdev, and M+P includes the perceptron stage on the same data.

### 4.2 Including out-of-domain data

It is not realistic to think that one can always obtain a whole training set adequate to the genre of the test set, but it may be feasible to collect a smaller set for development which shares the domain of the test. This can be enough to ease the domain shift. Let us assume now that we want to translate *N08* and we can use the help of *N06* for development together with *Trdev*. With both data sets we create three development sets: Trdev, N06 and Trdev.N06 (the union of both randomly sorted). These sets can be used for MERT alone or for the combined training with the perceptron.

Permutations of these three data sets in both stages provide nine different configurations. Figure 3 depicts the results on N08 for these nine models. The plot shows the BLEU evolution on test during the training and crosses mark the BLEU score at the stopping point of the perceptron. The initial point represents the score obtained after MER training and equals to 29.06 for Trdev set, 32.89 for N06 and 32.34 for Trdev.N06. The addition of N06 shows then crucial in improving the performance on N08 both for MERT and for the combined training. For MERT alone, N06 allows to increase more than 3 points the BLEU score on N08.

Regarding the effect of the perceptron training, it is still apparent when tuning the system towards the domain of N08. In all the models, the perceptron stage improves with statistical significance the final BLEU score, yet now improvements are small. More significant, the improvement comes in the highest range of the BLEU score. That is, a straightforward application of MERT on N08 achieves a BLEU of 32.77, and 7 out of the 9 combined trainings with the perceptron surpass this value, showing that MERT is not able to find the real minimum in this case, and that the generalisation of the weights using a similar set can take us close to an optimum result.

In all the cases there is a substantial improvement of the BLEU score already at the first steps of the training and it becomes stable after about 2,000 examples. For some models, the maximum is before this point as it happens for the best learnings. So, the stopping criterion based on the stabilisation of the weights is failing in these cases where the best BLEU is found very soon. However, notice that 6 out of the 9 models surpass the maximum value that one would obtain by MERT on N08 at least along 1,000 examples. That means that although not being able to catch the maximum BLEU, probably the selected score will still be better than the initial point.

Globally, one can see that, as expected, perceptron trainings which start from a better point (MERT with a similar set to the test one) reach faster and higher maxima. Perceptron trainings that base the learning on Trdev remain below those that use N06 or Trdev.N06. Numerically, the best results at the stopping point are obtained with the combinations $[\text{N06}]_{\text{M}}$+Trdev.N06 (33.08) and $[\text{N06}]_{\text{M}}$+N06 (33.07). The best attainable score with an ideal stopping criterion is a BLEU of 33.53 obtained with this last configuration, and it is 0.76 points over the forecasted upper bound that MERT could attain on N08. Notice that MERT alone also beats this value when trained with N06 ($[\text{N06}]_{\text{M}} \rightarrow 32.89$ vs. $[\text{N08}]_{\text{M}} \rightarrow 32.77$ both on N08). Still our best attainable score is 0.64 points over the best MERT one, although with our criterion we finally get a modest improvement of 0.19 BLEU points.

## 5 Discussion and conclusions

This work presents a combined methodology for estimating robust weights for the components of a

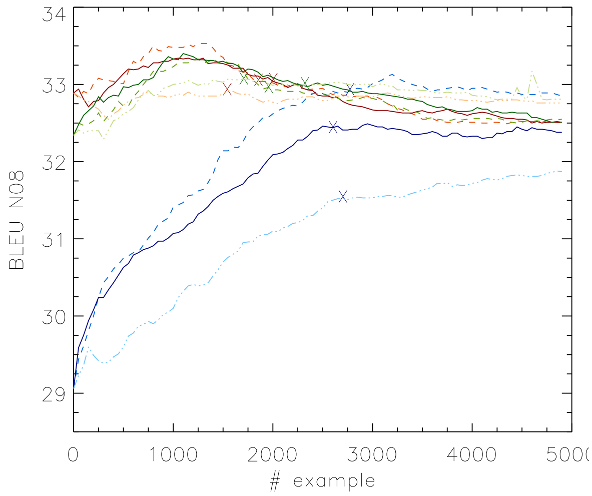


Figure 3: Evolution during the perceptron training of the BLEU score on the test set N08 for 9 different configurations. Initial points correspond to 3 different MERT results: $[\text{Trdev}]_{\text{M}}$ (blues), $[\text{N06}]_{\text{M}}$ (reds) and $[\text{Trdev.N06}]_{\text{M}}$ (greens). For each of them, the perceptron runs on Trdev (dot-dashed), N06 (dashed) and Trdev.N06 (solid). Crosses mark the point where one would stop the training on the development set.

log-linear model in SMT. The first stage is the traditional MERT. The second stage continues with a perceptron-based training that updates the value of each weight so that it is closer to the weight of the translation with the highest BLEU within an $n$-best list.

For an out-of-domain test set, N06 or N08, the perceptron training done with in-domain data improves by more than 2 BLEU points the value of MERT alone. The perceptron is able to cover two thirds of the distance between the MERT value and the fictitious score that MERT would obtain directly on these tests.

In the situation that one can collect a data set with a similar domain to the test set, both MERT and the perceptron training improve considerably the BLEU score. Even with these conditions, the best results are achieved when the out-of-domain data are used in the two stages of the combined training. A more precise stopping criteria could help to further enhance current results as seen in the evolution of the BLEU score along the whole training.

The initial feature vector for the percetron is also important for its performance. This is why the global best test results are obtained when using the best available MERT values. And this is also why we apply the perceptron training after MERT and not instead ot it.

As we have explained in the introduction, one can find in the literature several works that use a learning algorithm to estimate the vector of weights in SMT. A direct comparison with our results is difficult because of the different purposes of the analyses. However, we can point out the results of Chiang et al. (2008), where an improvement of 0.7 BLEU points on N06 is found with MIRA with respect to MERT for their same 12 features. They study the domain of the test set and realise that the improvement comes from the part of the set with web content instead of newswire, but attribute it to the length of the sentences, arguing that MIRA translations tend to be longer than the MERT ones and web references are longer than newswire's. On the other hand, Arun and Koehn (2007) obtain shorter outputs with MIRA for the Czech-English language pair. In our case, we also obtain longer translations with the perceptron than with MERT alone, probably because the experiments are done on the same language pair, Arabic-English. However, since the longer output is consistent for Trtest and N08 sets and we do not get any improvement on Trtest, we attribute the enhancements to the domain of the data set.

It might be interesting to extend this work from two points of view. From the empirical side, different corpus sizes and language pairs, above all those with data sets of several domains available, should be tested. The behaviour of the perceptron when one optimises against other metrics than BLEU should be studied as well. From the theoretical side, one must investigate which are the properties of the perceptron that allow to gain robustness in the weights. This may be useful since it is not clear how to characterise robust vectors in MERT (Foster and Kuhn, 2009).

## Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement numbers 247914 and 247762 and from the Spanish Ministry of Science and Innovation (TIN2009-14675-C03). The authors are thankful to the anonymous referees for some useful comments and suggestions.

## References


Arun, Abhishek and Philipp Koehn. 2007. Online learning methods for discriminative training of phrase based statistical machine translation. In *MT Summit XI*, pages 15–20, September.

Banerjee, Satanjeev and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*.

Cer, Daniel, Daniel Jurafsky, and Christopher Manning. 2008. Regularization and search for minimum error rate training. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 26–34, Columbus, Ohio, June.

Chiang, David, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 224–233, Morristown, NJ, USA.

Collins, Michael. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, volume 10, pages 1–8.

Doddington, George. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the 2nd Internation Conference on Human Language Technology*, pages 138–145.

Foster, George and Roland Kuhn. 2009. Stabilizing minimum error rate training. In *StatMT '09: Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 242–249, Morristown, NJ, USA. Association for Computational Linguistics.

Giménez, Jesús and Enrique Amigó. 2006. IQMT: A Framework for Automatic Machine Translation Evaluation. In *Procs. of the 5th Conference on Language Resources and Evaluation*, pages 685–690.

Giménez, Jesús. 2007. IQMT v 2.1. Technical Manual (LSI-07-29-R). Technical report, TALP Research Center. LSI Department. http://www.lsi.upc.edu/∼nlp/IQMT/IQMT.v2.1.pdf.

Koehn, Philipp, Wade Shen, Marcello Federico, Nicola Bertoldi, Chris Callison-Burch, Brooke Cowan, Chris Dyer, Hieu Hoang, Ondrej Bojar, Richard Zens, Alexandra Constantin, Evan Herbst, and Christine Moran. 2006. Open Source Toolkit for Statistical Machine Translation. Technical report, Johns Hopkins University Summer Workshop.

Koehn, Philipp, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computation Linguistics*, pages 177–180, Jun.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, Barcelona, Spain.

Liang, Percy, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 761–768, Morristown, NJ, USA.

Lin, Chin-Yew and Franz Josef Och. 2004. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

Moore, Robert C. and Chris Quirk. 2008. Random restarts in minimum error rate training for statistical machine translation. In *Proc. of the 22nd International Conference on Computational Linguistics*, pages 585–592, Manchester, UK, August.

Nießen, Sonja, Franz Josef Och, Gregor Leusch, and Hermann Ney. 2000. An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*.

Och, Franz Josef and Hermann Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the Association for Computational Linguistics*, Sapporo, Japan, July 6-7.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Association of Computational Linguistics*, pages 311–318.

Stolcke, Andreas. 2002. SRILM – An extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*.

Tillmann, Christoph and Tong Zhang. 2006. A discriminative global training algorithm for statistical mt. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 721–728, Sydney, Australia, July. Association for Computational Linguistics.