Online adaptation strategies for statistical machine translation in post-editing scenarios

Pascual Martínez-Gómez[a], Germán Sanchis-Trilles[b], Francisco Casacuberta[b]

[a]*Graduate School of Information Science and Technology*
Department of Computer Science
University of Tokyo
[b]*Departamento de Sistemas Informáticos y Computación*
Instituto Tecnológico de Informática
Universitat Politècnica de València

Abstract

One of the most promising approaches to *machine translation* consists in formulating the problem by means of a pattern recognition approach. By doing so, there are some tasks in which online adaptation is needed in order to adapt the system to changing scenarios. In the present work, we perform an exhaustive comparison of four online learning algorithms when combined with two adaptation strategies for the task of online adaptation in statistical machine translation. Two of these algorithms are already well-known in the pattern recognition community, such as the perceptron and passive-aggressive algorithms, but here they are thoroughly analyzed for their applicability in the statistical machine translation task. In addition, we also compare them with two novel methods, i.e., Bayesian predictive adaptation and discriminative ridge regression. In statistical machine translation, the most successful approach is based on a log-linear approximation to a posteriori distribution. According to experimental results, adapting the scaling factors of this log-linear combination of models using discriminative ridge regression or

Bayesian predictive adaptation yields the best performance.

Keywords: online learning, adaptation, statistical machine translation

1. Introduction

With the increase of both manually annotated data and computational resources, pattern recognition techniques (PR) have evolved to become state-of-the-art in tasks that have been historically reserved for humans due to having a highly structured output and a high level of ambiguity. Research fields such as speech recognition, machine translation, or image annotation have experienced an important breakthrough as a result of embracing PR. Although these systems typically perform well on tasks that are similar to the one that they have been trained on, performance decreases abruptly when the task becomes slightly different [1, 2].

In tasks where supervised learning is required, obtaining manually annotated corpora for every specific domain might not be realistic. Thus, adaptation techniques are strongly in demand to deal with the lack of domain-specific data. In addition, some tasks require the system to adapt itself after every observation is presented to the system. Since a complete retraining is often unfeasible, online learning techniques [3] are often embraced, leading to *online adaptation* [4].

Online adaptation is a problem that is currently present in a variety of research fields like speech recognition [5] or image recognition [6]. In this work, different approaches to online adaptation are studied within the specific task of *statistical machine translation* (SMT). Adapting a system to changing tasks is particularly interesting in the *computer assisted translation* (CAT) [7] and *interactive machine translation* (IMT) [8] paradigms, where collaboration between human translators and machine translation systems is essential in producing high quality results ef-

ficiently. In these scenarios, the SMT system proposes a hypothesis to a human translator, who may amend the hypothesis to obtain an acceptable target sentence. The human translator then expects the system to learn dynamically from its own errors so that the errors that were corrected once do not need to be corrected again. Furthermore, it is often the case that human translators need to translate many documents with different styles and topics in limited time. The challenge is then to make the best use of every correction provided by the user by adapting the models *online*, i.e., without a complete retraining of the model parameters, since this retraining might not be feasible with the user actively waiting for the system's output. Experiments show that significant improvements can be achieved by means of online learning, which would lead to a reduction in the time and effort that a human translator would need to correct the system hypotheses.

The foundation for modern SMT, the pattern recognition approach to machine translation, was established in [9], by formulating the SMT problem as follows: given an input sentence $\mathbf{x}$ in a certain source language, the best translation $\hat{\mathbf{y}}$ in a certain target language is to be found, maximizing the posterior probability:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \; p(\mathbf{y} \mid \mathbf{x}). \tag{1}$$

Current state-of-the-art SMT systems find the best translation of $\mathbf{x}$ by modeling the posterior probability $p(\mathbf{y} \mid \mathbf{x})$ directly by means of the so-called log-linear models [10], where the decision rule is given by

$$\begin{aligned} \hat{\mathbf{y}} &= \underset{\mathbf{y}}{\operatorname{argmax}} \frac{\exp \sum_{m=1}^{M} \lambda_m h_m(\mathbf{x}, \mathbf{y})}{\sum_{y'} \exp \sum_{m=1}^{M} \lambda_m h_m(\mathbf{x}, \mathbf{y}')} \\ &= \underset{\mathbf{y}}{\operatorname{argmax}} \sum_{m=1}^{M} \lambda_m h_m(\mathbf{x}, \mathbf{y}) \\ &= \underset{\mathbf{y}}{\operatorname{argmax}} \boldsymbol{\lambda} \cdot \mathbf{h}(\mathbf{x}, \mathbf{y}) = \underset{\mathbf{y}}{\operatorname{argmax}} s(\mathbf{x}, \mathbf{y}). \end{aligned} \tag{2}$$

Here $h_m(\mathbf{x}, \mathbf{y})$ is a score function that represents an important feature for the translation of $\mathbf{x}$ into $\mathbf{y}$, M is the number of models (or features), and λ_m are the weights that act as scaling factors of the score functions. $s(\mathbf{x}, \mathbf{y})$ represents the score of a hypothesis $\mathbf{y}$ given an input sentence $\mathbf{x}$ and is not treated as a probability since the normalization term has been omitted. Common feature functions $h_m(\mathbf{x}, \mathbf{y})$ not only include different translation models (TM) that describe the correspondence in words or sequences of words between languages, they also include distortion models that account for necessary reorderings of blocks of words and language models that account for the well-formedness of the translated sentence. Feature functions defined as probabilities are usually specified in the logarithmic domain, which is why Eq. 2 is said to be a log-linear model. However, some of the feature functions used in modern SMT systems do not describe probabilities (e.g. the length of $\mathbf{y}$). Here, we only attempt to adapt a subset of the feature functions that describe probabilities, namely those defined at the local translation unit level.

Typically, $\mathbf{h}(\mathbf{x} \mid \mathbf{y})$ and $\boldsymbol{\lambda}$ are estimated by means of training and development sets, respectively. This leads to one important problem in SMT: whenever this data belongs to a different domain than the text to be translated, the translation quality diminishes significantly [2]. Hence, the adaptation problem is very common in SMT, where the goal is to improve the performance of systems trained and tuned on out-of-domain data by using limited amounts of in-domain data.

Originally, SMT systems relied on word-to-word translations, in which each source word was translated independently and then reordered. This approach had the important drawback of not being able to cope with context, and hence failed when attempting to translate multi-word expressions. For this reason, *phrase-based* (PB) models [9] were introduced, widely outperforming single word mod-

els [2]. The basic idea of PB translation is to segment $\mathbf{x}$ into *phrases* (i.e., word sequences), then to translate each source phrase $\tilde{x}_k \in \mathbf{x}$ into a target phrase $\tilde{y}_k \in \mathbf{y}$, and finally reorder them to compose the target sentence $\mathbf{y}$. Typically, bilingual phrase pairs obtained at training time are stored in a huge table (often referred to as *phrase-table*) along with different features that can be defined at the local phrase level. PB models have been employed throughout this article.

The rest of this paper is structured as follows. Section 2 presents a short review of the current work in adaptation and online learning in different research fields. In Section 3, two different strategies for incorporating online learning capabilities into a CAT system are presented. These strategies adapt either the scaling factors $\boldsymbol{\lambda}$ or the feature functions $\mathbf{h}$. In Section 4, two online learning algorithms that are available in the literature are instantiated here for the purpose of adaptation in a CAT environment. In addition, two new algorithms are proposed for the specific problem dealt with here. The experimental setup and empirical results are presented in Section 5. The last section details conclusions and future work.

2. Related work

Batch adaptation (as opposed to online) is a very broad area that has received a large amount of attention in different fields. In [1], the maximum likelihood framework is applied to speaker adaptation. In [11], the maximum likelihood framework is expanded, obtaining maximum a posteriori estimators that are aimed at adapting model parameters. In [12], adaptation is confronted as a classification problem by extending the set of features with a domain-specific tag.

However, there are also cases where there is no adaptation data at all available beforehand, and the system needs to adapt itself online without falling into an

excessive time burden. This problem, among others, has lead to the development of an incremental version of the Expectation-Maximization algorithm [13]. This algorithm has been successfully applied in an IMT scenario in [14], where the models involved are incrementally updated as the user feedback is received.

The perceptron algorithm is probably one of the most popular online learning algorithms in the machine learning field [15], where the parameters are updated after the label of the current observation is presented to the system. Therefore, this algorithm has been included in the present study for comparison purposes.

The passive-aggressive (PA) framework [16] is a popular family of margin-based online learning algorithms. In [17], the authors propose the use of the PA framework to update the feature functions $\mathbf{h}$. The improvements obtained were very limited, since adapting $\mathbf{h}$ is a very sparse problem. For this reason, we compare adapting $\mathbf{h}$ with adapting $\boldsymbol{\lambda}$, which is shown in [18] to be a good adaptation strategy. In [18], the scaling factors are adapted by means of an adaptation set in a Bayesian learning fashion. In contrast, our purpose is to perform online adaptation, i.e., to adapt system parameters after each new sample has been provided.

Several works make use of online learning algorithms for learning the scaling factors in SMT. However, most of them employ these algorithms for the purpose of training phrase-based systems discriminatively and boosting the number of features present. This is the case in [19], where a perceptron-style algorithm is used to learn $\boldsymbol{\lambda}$, and in [20, 21], where online large-margin algorithms are used for that same purpose. Both approaches are then compared in [22].

In this work, we present an in-depth comparison of four online adaptation algorithms, i.e., passive-aggressive, perceptron, discriminative ridge regression, and Bayesian predictive adaptation. The passive-aggressive algorithm and the percep-

tron algorithm have been applied by other authors to feature function and scaling factor adaptation, respectively. In this work, in order to perform a meaningful comparison, the passive-aggressive algorithm has been applied to scaling factor adaptation and the perceptron has also been used for feature function adaptation. The authors of the present article have recently applied Bayesian predictive adaptation and discriminative ridge regression to scaling factor adaptation. The application to feature function adaptation is also shown here for completeness. All these algorithms have been applied to feature function and scaling factor adaptation within a simulated CAT environment, in which the SMT system receives the correct translation after having produced its own automatic translation.

3. Adaptation approaches

In general, in an online learning framework, the learning algorithm processes observations sequentially. After every input, the system makes a prediction and then receives feedback, which can range from a simple opinion of how good the system's prediction was to the true label of the input in completely supervised environments. The purpose of online learning is to modify the prediction mechanisms in order to improve the quality of future decisions. Specifically, in a CAT scenario, the SMT system receives a source sentence and then outputs a translation hypothesis. The user then post-edits the system's hypothesis, producing a reference translation $\mathbf{y}^{\tau}$ that can be used as supervised feedback. The purpose is to learn from that interaction. Thus, Eq. 2 is redefined as follows

$$\begin{aligned} \hat{\mathbf{y}}_t &= \underset{\mathbf{y}}{\text{argmax}} \sum_{m=1}^{M} \lambda_m^t h_m^t(\mathbf{x}_t, \mathbf{y}) \\ &= \underset{\mathbf{y}}{\text{argmax}}\, \boldsymbol{\lambda}^t \cdot \mathbf{h}^t(\mathbf{x}_t, \mathbf{y}), \end{aligned} \tag{3}$$

where feature functions $\mathbf{h}^t$ and log-linear weights $\boldsymbol{\lambda}^t$ vary according to samples $(\mathbf{x}_1, \mathbf{y}_1^\tau), \ldots, (\mathbf{x}_{t-1}, \mathbf{y}_{t-1}^\tau)$ seen before time t. In order to simplify notation, we will omit subindex t from input sentence $\mathbf{x}$ and output sentence $\hat{\mathbf{y}}$, although it is always assumed. We can apply online adaptation to either $\mathbf{h}^t$ or $\boldsymbol{\lambda}^t$, or to both at the same time. However, in this paper, we focus on adapting only one at a time.

The hypothesis $\hat{\mathbf{y}}$ that maximizes the likelihood is not necessarily the hypothesis with the highest quality from a human perspective or in terms of a certain quality measure. Let $\mathbf{y}^*$ be the hypothesis with the highest quality, but which might have a lower likelihood[1]. Our purpose is to adapt the model parameters so that $\mathbf{y}^*$ is rewarded and achieves a higher score according to Eq. 3.

We define the *difference* in translation quality between the proposed hypothesis $\hat{\mathbf{y}}$ and the best hypothesis $\mathbf{y}^*$ in terms of a given quality measure $\mu(\cdot)$:

$$l(\hat{\mathbf{y}}) = |\mu(\hat{\mathbf{y}}) - \mu(\mathbf{y}^*)|, \tag{4}$$

where the absolute value has been introduced in order to preserve generality, since in SMT some of the quality measures used, such as TER [23], represent an error rate (i.e., the lower the better), whereas others such as BLEU [24] measure precision (i.e., the higher the better). The term $l(\hat{\mathbf{y}})$ also depends on the best hypothesis $\mathbf{y}^*$ and, in turn, also depends on sentence $\mathbf{x}$ and reference translation $\mathbf{y}^\tau$. Those dependencies are not explicitly included in order to keep notation uncluttered. The score difference between $\hat{\mathbf{y}}$ and $\mathbf{y}^*$ is related to $\phi(\hat{\mathbf{y}})$, which is defined as

$$\phi(\hat{\mathbf{y}}) = s(\mathbf{x}, \mathbf{y}^*) - s(\mathbf{x}, \hat{\mathbf{y}}), \tag{5}$$

where again the dependencies with $\mathbf{x}$, $\mathbf{y}^\tau$, and $\mathbf{y}^*$ have been omitted to simplify notation. Ideally, we would like differences in $l(\cdot)$ to correspond to differences in

[1] $\mathbf{y}^*$ does not necessarily match the reference translation $\mathbf{y}^\tau$ due to eventual coverage problems.

$\phi(\cdot)$: if hypothesis $\mathbf{y}$ has a translation quality $\mu(\mathbf{y})$ that is very similar to the translation quality of $\mu(\mathbf{y}^*)$, we would like this to be reflected in translation score s, i.e., $s(\mathbf{x}, \mathbf{y})$ is very similar to $s(\mathbf{x}, \mathbf{y}^*)$. Hence, the purpose of our online procedure should be to promote this correspondence after each sample $(\mathbf{x}_t, \mathbf{y}_t^\tau)$.

3.1. Scaling factor adaptation

A coarse-grained technique for tackling the online learning problem in SMT implies adapting the log-linear scaling factors $\boldsymbol{\lambda}$ present in Eq. 3. In order to compute the new scaling factors $\boldsymbol{\lambda}^t$, the previously learned $\boldsymbol{\lambda}^{t-1}$ needs to be combined with an appropriate update step $\check{\boldsymbol{\lambda}}^t$. The aim is to compute an appropriate update term $\check{\boldsymbol{\lambda}}^t$ for translating the sentence pair observed at time t, $(\mathbf{x}_t, \mathbf{y}_t^\tau)$, and then obtain $\boldsymbol{\lambda}^t$. This is often done as a linear combination [25], where

$$\boldsymbol{\lambda}^t = (1 - \alpha)\boldsymbol{\lambda}^{t-1} + \alpha\check{\boldsymbol{\lambda}}^t, \tag{6}$$

for a certain learning rate α. Computing $\boldsymbol{\lambda}^t$ can be seen as a rudimentary predictor-corrector step where estimation $\boldsymbol{\lambda}^{t-1}$ is corrected by an appropriate update step $\check{\boldsymbol{\lambda}}^t$.

The information that is taken into account when computing $\check{\boldsymbol{\lambda}}^t$ is general and imprecise, but the variation in score in Eq. 3 can be high since the scaling factors of the log-linear model will be modified: when adapting the system to a new domain, the importance of every single model will be adjusted in an online manner.

3.2. Feature function adaptation

As discussed in Section 1, state-of-the-art SMT systems present a log-linear combination of feature functions $h_m(\mathbf{x}, \mathbf{y})$. These feature functions include several translation models, such as $p(\mathbf{x} \mid \mathbf{y})$ and $p(\mathbf{y} \mid \mathbf{x})$. They also include models to cope with word-reorderings between the source language and the target language,

as well as the target language model which assesses how well-formed the target sentence is. Typically, the translation models are stored in a phrase-table since they can be defined at the phrase level, i.e., $h_m(\mathbf{x} \mid \mathbf{y}) = \sum_k h_m(\tilde{x}_k \mid \tilde{y}_k)$ when working in the logarithmic domain. However, the rest of the models cannot decompose into phrase-specific scores because of their nature. Instead of attempting to adapt all of the feature functions, we will only research the online adaptation of the translation models. For this purpose, we will define $h_s(\mathbf{x}, \mathbf{y})$ as the combination of all translation models defined at the phrase level, and adaptation will be performed only for h_s. The reason for adapting only h_s (and not the individual models or even the other feature functions) is that, even in this case, the number of parameters to be adapted is on the order of several million. Let B be the size of the phrase-table (i.e., several million elements). We will denote by $\mathbf{g}(\tilde{x}, \tilde{y})$ a function returning a vector of B components, whose entries are all zero except for the one corresponding to phrase pair $(\tilde{x}, \tilde{y})$, whose value is the score of $h_s(\tilde{x}, \tilde{y})$ for that specific phrase pair. $\mathbf{g}(\mathbf{x}, \mathbf{y})$ will return a vector of size B, with all entries set to zero except those of all phrase pairs $(\tilde{x}_k, \tilde{y}_k)$ that build up sentence pair $(\mathbf{x}, \mathbf{y})$.

Even though parameters within $h_s(\mathbf{x}, \mathbf{y})$ could be altered directly, it is technically more straightforward to introduce an auxiliary vector $\mathbf{u} \in \mathbb{R}^B$, following the work in [17, 26] and for purposes of comparison. Then, the value of $h_s(\mathbf{x}, \mathbf{y})$ at time t can be expressed as a product of two vectors such that

$$h_s^t(\mathbf{x}, \mathbf{y}) = \mathbf{u}^{t-1} \cdot \mathbf{g}(\mathbf{x}, \mathbf{y}). \tag{7}$$

Note that the entries of the auxiliary vector $\mathbf{u}$ are initially all set to 1 (at $t = 0$).

As done when updating $\boldsymbol{\lambda}$, $\mathbf{u}$ will be updated following a linear combination:

$$\mathbf{u}^t = (1 - \alpha)\mathbf{u}^{t-1} + \alpha\breve{\mathbf{u}}^t. \tag{8}$$

Here $\mathbf{u}^{t-1}$ is the vector that has been learned after observing the previous $(\mathbf{x}_1, \mathbf{y}_1^\tau), \ldots, (\mathbf{x}_{t-1}, \mathbf{y}_{t-1}^\tau)$ sentence pairs, and $\check{\mathbf{u}}^t$ is the online update after observing the t-th sample $(\mathbf{x}_t, \mathbf{y}_t^\tau)$. Note that $\check{\mathbf{u}}^t$ does not need to be the optimum $\mathbf{u}$ at time t, but only the update step, which may even be just a gradient.

Since only h_s varies after each new observation, Eq. 3 can be redefined as

$$\hat{\mathbf{y}}_t = \underset{\mathbf{y}}{\operatorname{argmax}} \sum_{m \notin s} \lambda_m h_m(\mathbf{x}, \mathbf{y}) + h_s^t(\mathbf{x}, \mathbf{y}) = \underset{\mathbf{y}}{\operatorname{argmax}}\, h_r(\mathbf{x}, \mathbf{y}) + h_s^t(\mathbf{x}, \mathbf{y}), \quad (9)$$

where s denotes here the set of all features h_m which can be defined at the local phrase level, and $h_r(\mathbf{x}, \mathbf{y})$ is the combination of all features which are not defined at the phrase level, i.e., all the features except the translation models. Note that h_r does not have a super-index t because it is constant in time.

4. Online methods

In this section, passive-aggressive, perceptron, discriminative ridge regression, and Bayesian predictive adaptation are introduced. The general philosophy for each method is described first. Then, the application to every adaptation strategy is presented. Sub- and super-indices t might be omitted for clarity if the context is obvious or stated explicitly when the temporal relation requires a clear distinction.

4.1. Passive-aggressive

Passive-aggressive (PA) [16] is a family of margin-based, on-line learning algorithms that update model parameters after each new observation has been seen. In this case, PA is applied to a regression problem, where a target value has to be predicted by the system for the hypothesis $\mathbf{y}$ at time t by using a linear regression function where the weights are the parameters that have to be learned.

After every prediction, the true target value is revealed and the system suffers an instantaneous loss. If the error made falls below a certain sensitivity parameter, the loss suffered by the system is zero and the algorithm remains *passive*, i.e., the new weight vector is equal to the previous one. Otherwise, the loss grows linearly with the error and the algorithm *aggressively* forces an update of the parameters. The idea behind the PA algorithm is to compute the weights of the regression function so that it achieves a zero loss function on the current input while remaining as close as possible to the previous weight vector.

4.1.1. Passive-aggressive for scaling factor adaptation

The constrained optimization problem yielding update term $\check{\boldsymbol{\lambda}}^t$ is stated as [16]:

$$\check{\boldsymbol{\lambda}}^t = \underset{\boldsymbol{\lambda}}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{\lambda} - \boldsymbol{\lambda}^{t-1}||^2 + C\xi^2 \qquad \text{s.t.} \;\; l(\hat{\mathbf{y}}) = 0, \tag{10}$$

where ξ^2 is a squared slack variable that is scaled by the aggressivity factor C, according to the so-called PA Type-II. $l(\hat{\mathbf{y}})$ is the difference in translation quality between the hypothesis proposed by the system and the best hypothesis as described in Eq. 4. It is common to add a slack variable into the optimization problem to achieve more flexibility during the learning process. Once the optimization problem is defined, the update term may be obtained by adding the constraint together with a Lagrangian variable and setting the partial derivatives to zero [17]:

$$\check{\boldsymbol{\lambda}}^t = \Phi(\hat{\mathbf{y}}) \frac{\sqrt{l(\hat{\mathbf{y}})} - \boldsymbol{\lambda}^{t-1}\Phi(\hat{\mathbf{y}})}{||\Phi(\hat{\mathbf{y}})||^2 + \frac{1}{C}}, \tag{11}$$

where $\Phi(\hat{\mathbf{y}}) = \mathbf{h}(\mathbf{x}, \mathbf{y}^*) - \mathbf{h}(\mathbf{x}, \hat{\mathbf{y}})$, and the update is triggered when $\boldsymbol{\lambda}^{t-1}\Phi(\hat{\mathbf{y}}) \geq \sqrt{l(\hat{\mathbf{y}})}$ is violated. Plugging $\check{\boldsymbol{\lambda}}^t$ into Eq. 6 results in a generalization of PA that matches the original work in [16] when $\alpha = 1$. Figure 1 shows the pseudo-code for the PA-II online adaptation algorithm when adapting scaling factors $\boldsymbol{\lambda}$.

while input sentences $\mathbf{x}_t$ in stream **do**

$\hat{\mathbf{y}} \leftarrow \underset{\mathbf{y}}{\text{argmax}}\, \boldsymbol{\lambda}^{t-1} \cdot \mathbf{h}^t(\mathbf{x}_t, \mathbf{y})$

output $\hat{\mathbf{y}}$

if not $\boldsymbol{\lambda}^{t-1}\Phi(\hat{\mathbf{y}}) \geq \sqrt{l(\hat{\mathbf{y}})}$ **then**

$\Phi(\hat{\mathbf{y}}) \leftarrow \mathbf{h}(\mathbf{x}_t, \mathbf{y}^*) - \mathbf{h}(\mathbf{x}_t, \hat{\mathbf{y}})$

$\check{\boldsymbol{\lambda}}^t \leftarrow \Phi(\hat{\mathbf{y}}) \frac{\sqrt{l(\hat{\mathbf{y}})} - \boldsymbol{\lambda}^{t-1}\Phi(\hat{\mathbf{y}})}{||\Phi(\hat{\mathbf{y}})||^2 + \frac{1}{C}}$

$\boldsymbol{\lambda}^t \leftarrow (1-\alpha)\boldsymbol{\lambda}^{t-1} + \alpha\check{\boldsymbol{\lambda}}^t$

else $\boldsymbol{\lambda}^t \leftarrow \boldsymbol{\lambda}^{t-1}$

end if

end while

Figure 1: Pseudo-code for PA-II online learning of the scaling factors $\boldsymbol{\lambda}$ as described in Section 4.1.

4.1.2. Passive-aggressive for feature function adaptation

The constrained optimization can be similarly formulated for feature function adaptation as:

$$\check{\mathbf{u}}^t = \underset{\mathbf{u}}{\text{argmin}}\, \frac{1}{2}||\mathbf{u} - \mathbf{u}^{t-1}||^2 + C\xi^2 \qquad \text{s.t.} \;\; l(\hat{\mathbf{y}}) = 0, \tag{12}$$

and the solution to this problem is given by the expression of the update term

$$\check{\mathbf{u}}^t = \Phi(\hat{\mathbf{y}}) \frac{\sqrt{l(\hat{\mathbf{y}})} - \mathbf{u}^{t-1}\Phi(\hat{\mathbf{y}})}{||\Phi(\hat{\mathbf{y}})||^2 + \frac{1}{C}}, \tag{13}$$

where $\Phi(\hat{\mathbf{y}}) = \mathbf{g}(\mathbf{x}, \mathbf{y}^*) - \mathbf{g}(\mathbf{x}, \hat{\mathbf{y}})$, with $\mathbf{g}(\cdot,\cdot)$ being defined in Eq. 7. As in [17], the update is triggered only when $\hat{\mathbf{y}}$ violates the constraint $\mathbf{u}^{t-1}\Phi(\hat{\mathbf{y}}) \geq \sqrt{l(\hat{\mathbf{y}})}$.

4.1.3. Heuristic variation

Another update condition that is different to the one described in the previous section has been explored in this work. In this variation, an update always has to be performed whenever the quality of a predicted hypothesis $\hat{\mathbf{y}}$ is worse than the quality of the best possible hypothesis $\mathbf{y}^*$, in terms of quality measure $\mu(\cdot)$:

$$\exists \mathbf{y}^* : |\mu(\mathbf{y}^*) - \mu(\hat{\mathbf{y}})| > 0. \tag{14}$$

In this way, parameter estimations that lead to better hypotheses are rewarded. Symmetrically, estimations that produce lower quality hypotheses are penalized.

4.2. Perceptron

A standard single-layer perceptron has a number of inputs whose dimensionality depends on the dimension of the observations. Every variable of a given observation is weighted and combined in its hidden layer with the rest of the variables to produce a single output. As an error-driven algorithm, the output produced is compared with the ideal target value and the parameters of the perceptron are adjusted to reduce the difference between the target and the output values.

Although the perceptron algorithm [15, 27] is very popular, a simple variation, the perceptron-like algorithm (PCL), is also widely used [28, 29, 30]. Among the algorithms studied in this work, PCL is the simplest and fastest method since it involves only three operations on vectors. PCL was used instead of the standard perceptron for comparison reasons since PCL was successfully applied in [28] and outperformed the regular perceptron in preliminary experimentation. Specifically, the PCL algorithm makes a fixed-length shift of the previous weight vector in the direction of the difference (gradient) between the output produced ($\hat{\mathbf{y}}$) and the output that the system should have produced ($\mathbf{y}^*$).

4.2.1. Perceptron for scaling factor adaptation

To adapt the scaling factors, the input sample for the perceptron is the value of $\mathbf{h}(\mathbf{x}_t, \hat{\mathbf{y}}_t)$ at each time t. These samples are M-dimensional, where M is the total number of models (typically around 14). Those vectors are combined linearly with the scaling factors $\boldsymbol{\lambda}$ so that the shift term $\check{\boldsymbol{\lambda}}^t$ is computed for PCL as [28]

$$\check{\boldsymbol{\lambda}}^t = \mathrm{sign}(\mathbf{h}(\mathbf{x}, \mathbf{y}^*) - \mathbf{h}(\mathbf{x}, \hat{\mathbf{y}})). \tag{15}$$

while input sentences $\mathbf{x}_t$ in stream **do**

$\hat{\mathbf{y}} \leftarrow \underset{\mathbf{y}}{\text{argmax}}\, \boldsymbol{\lambda}^{t-1} \cdot \mathbf{h}^t(\mathbf{x}_t, \mathbf{y})$

return $\hat{\mathbf{y}}$

$\check{\boldsymbol{\lambda}}^t \leftarrow \text{sign}(\mathbf{h}(\mathbf{x}_t, \mathbf{y}^*) - \mathbf{h}(\mathbf{x}_t, \hat{\mathbf{y}}))$

$\boldsymbol{\lambda}^t \leftarrow (1-\alpha)\boldsymbol{\lambda}^{t-1} + \alpha\check{\boldsymbol{\lambda}}^t$

end while

Figure 2: Pseudo-code for PCL online learning of the scaling factors $\boldsymbol{\lambda}$ as described in Section 4.2.

In this perceptron-like algorithm, the sign operator implies that fixed-size steps will be taken when adapting the parameters. This can be seen as a watered-down version of the standard and widely used perceptron whose steps might be, in its most general form, calibrated by the difference between desired output and actual output. Figure 2 shows the pseudo-code for PCL when adapting $\boldsymbol{\lambda}$.

4.2.2. Perceptron for feature function adaptation

To adapt $\mathbf{h}$, the necessary perceptron will be much larger since the input samples are feature vectors $\mathbf{g}(\mathbf{x}, \cdot)$, which are then combined linearly in the single layer of the perceptron with the weights of the auxiliary function $\mathbf{u}^{t-1}$.

Using the feature vector $\mathbf{g}(\mathbf{x}, \hat{\mathbf{y}})$ of the system's hypothesis $\hat{\mathbf{y}}$ and the feature vector $\mathbf{g}(\mathbf{x}, \mathbf{y}^*)$ of the best possible hypothesis $\mathbf{y}^*$ from the system, the PCL shift term for adapting $\mathbf{h}$ is computed in the same way as for the case of $\check{\boldsymbol{\lambda}}^t$:

$$\check{\mathbf{u}}^t = \text{sign}(\mathbf{g}(\mathbf{x}, \mathbf{y}^*) - \mathbf{g}(\mathbf{x}, \hat{\mathbf{y}})). \tag{16}$$

4.3. Discriminative ridge regression

PA and PCL algorithms try to find a configuration of the weight vectors such that *good* hypotheses tend to score *higher*. Discriminative regression, however, also enforces that *bad* hypotheses score *lower*, by using all hypotheses within a

given N-best list. Here, we present a method which we have named *discriminative ridge regression* (DRR) [31], which uses the ridge regression technique to develop a discriminative online adaptation algorithm.

4.3.1. DRR for scaling factor adaptation

The DRR algorithm requires an N-best list of hypotheses in decreasing order of likelihood. Let $nbest(\mathbf{x})$ be such a list computed by our models for sentence $\mathbf{x}$. To adapt $\boldsymbol{\lambda}$, we define an $N \times M$ matrix $\mathrm{H}_{\mathbf{x}}$ that contains the feature functions $\mathbf{h}$ of every hypothesis, where M is the number of features in Eq. 3:

$$\mathrm{H}_{\mathbf{x}} = [\mathbf{h}(\mathbf{x}, \mathbf{y}_1), \ldots, \mathbf{h}(\mathbf{x}, \mathbf{y}_N)]' \,. \tag{17}$$

Additionally, let $\mathrm{H}^*_{\mathbf{x}}$ be a matrix such that

$$\mathrm{H}^*_{\mathbf{x}} = [\mathbf{h}(\mathbf{x}, \mathbf{y}^*), \ldots, \mathbf{h}(\mathbf{x}, \mathbf{y}^*)]' \,, \tag{18}$$

where all rows are identical and equal to the feature vector of the best hypothesis $\mathbf{y}^*$ within the N-best list. Then, $\mathrm{R}_{\mathbf{x}}$ is defined as

$$\mathrm{R}_{\mathbf{x}} = \mathrm{H}^*_{\mathbf{x}} - \mathrm{H}_{\mathbf{x}} \,. \tag{19}$$

The key idea is to find a vector $\check{\boldsymbol{\lambda}}^t$ such that differences in scores are reflected as differences in the quality of the hypotheses. That is,

$$\mathrm{R}_{\mathbf{x}} \cdot \check{\boldsymbol{\lambda}}^t \propto \mathbf{l}_{\mathbf{x}} \,, \tag{20}$$

where $\mathbf{l}_{\mathbf{x}}$ is a column vector of N rows such that $\mathbf{l}_{\mathbf{x}} = [l(\mathbf{y}_1) \ldots l(\mathbf{y}_i) \ldots l(\mathbf{y}_N)]'$, $\forall \mathbf{y}_i \in nbest(\mathbf{x})$. The objective is to find $\check{\boldsymbol{\lambda}}^t$ such that

$$\check{\boldsymbol{\lambda}}^t = \operatorname*{argmin}_{\boldsymbol{\lambda}} |\mathrm{R}_{\mathbf{x}} \cdot \boldsymbol{\lambda} - \mathbf{l}_{\mathbf{x}}| \tag{21}$$

$$= \operatorname*{argmin}_{\boldsymbol{\lambda}} ||\mathrm{R}_{\mathbf{x}} \cdot \boldsymbol{\lambda} - \mathbf{l}_{\mathbf{x}}||^2, \tag{22}$$

while input sentences $\mathbf{x}_t$ in stream **do**

$\hat{\mathbf{y}} \leftarrow \underset{\mathbf{y}}{\text{argmax}}\, \boldsymbol{\lambda}^{t-1} \cdot \mathbf{h}^t(\mathbf{x}_t, \mathbf{y})$

output $\hat{\mathbf{y}}$

$\mathrm{H}_{\mathbf{x}_t} \leftarrow [\mathbf{h}(\mathbf{x}_t, \mathbf{y}_1), \ldots, \mathbf{h}(\mathbf{x}_t, \mathbf{y}_N)]'$, $\forall \mathbf{y}_i \in nbest(\mathbf{x}_t)$

$\mathrm{H}^*_{\mathbf{x}_t} \leftarrow [\mathbf{h}(\mathbf{x}_t, \mathbf{y}^*), \ldots, \mathbf{h}(\mathbf{x}_t, \mathbf{y}^*)]'$

$\mathrm{R}_{\mathbf{x}_t} \leftarrow \mathrm{H}^*_{\mathbf{x}_t} - \mathrm{H}_{\mathbf{x}_t}$

$\check{\boldsymbol{\lambda}}^t \leftarrow \left(\mathrm{R}'_{\mathbf{x}_t} \cdot \mathrm{R}_{\mathbf{x}_t} + \beta \mathrm{I}\right)^{-1} \mathrm{R}'_{\mathbf{x}_t} \cdot \mathbf{l}_{\mathbf{x}_t}$

$\boldsymbol{\lambda}^t \leftarrow (1-\alpha)\boldsymbol{\lambda}^{t-1} + \alpha \check{\boldsymbol{\lambda}}^t$

end while

Figure 3: Pseudo-code for DRR online learning of the scaling factors $\boldsymbol{\lambda}$ as described in Section 4.3.

where $||\cdot||^2$ is the Euclidean norm. Although Eqs. 21 and 22 are equivalent (i.e., the $\hat{\boldsymbol{\lambda}}$ that minimizes the first one also minimizes the second one), Eq. 22 allows for a direct implementation thanks to the ridge regression[2]. $\check{\boldsymbol{\lambda}}^t$ can be computed as the solution to the overdetermined system $\mathrm{R}_{\mathbf{x}} \cdot \check{\boldsymbol{\lambda}}^t = \mathbf{l}_{\mathbf{x}}$, which is given by

$$\check{\boldsymbol{\lambda}}^t = \left(\mathrm{R}'_{\mathbf{x}} \cdot \mathrm{R}_{\mathbf{x}} + \beta \mathrm{I}\right)^{-1} \mathrm{R}'_{\mathbf{x}} \cdot \mathbf{l}_{\mathbf{x}} \,, \tag{23}$$

where a small β is used as a regularization term to stabilize $\mathrm{R}'_{\mathbf{x}} \cdot \mathrm{R}_{\mathbf{x}}$ and to ensure that it is invertible. Figure 3 shows the pseudo-code for DRR when adapting $\boldsymbol{\lambda}$.

4.3.2. DRR for feature function adaptation

To adapt $\mathbf{h}$, the rows of matrix $\mathrm{H}_{\mathbf{x}}$ are the feature vectors $\mathbf{g}$ that correspond to the combination of TMs for every hypothesis. Hence, we will denote $\mathrm{H}_{\mathbf{x}}$ as $\mathrm{G}_{\mathbf{x}}$:

$$\mathrm{G}_{\mathbf{x}} = [\mathbf{g}(\mathbf{x}, \mathbf{y}_1), \ldots, \mathbf{g}(\mathbf{x}, \mathbf{y}_N)]' \,. \tag{24}$$

Note that the dimension of $\mathrm{G}_{\mathbf{x}}$ can get very large in this case since it is a $N \times B$ matrix, with B being in the range of millions of elements.

[2]Also known as Tikhonov regularization.

Then, the score for all hypotheses is represented by the column vector

$$\mathbf{s}_{\mathbf{x}} = \mathrm{H}_r \cdot \boldsymbol{\lambda}_m + \mathrm{G}_{\mathbf{x}} \cdot \mathbf{u}^{t-1} \,, \tag{25}$$

where, following the notation in Eq. 9, H_r is an $N \times |m \notin TM|$ matrix that contains all models that are not part of the TMs, i.e., language or reordering models, for every $\mathbf{y} \in nbest(\mathbf{x})$. $\boldsymbol{\lambda}_m$ are the corresponding scaling factors for H_r.

The definition of $\mathrm{G}^*_{\mathbf{x}}$ for feature function adaptation is

$$\mathrm{G}^*_{\mathbf{x}} = [\mathbf{g}(\mathbf{x}, \mathbf{y}^*), \ldots, \mathbf{g}(\mathbf{x}, \mathbf{y}^*)]' \,, \tag{26}$$

where all rows are identical and equal to the features of hypothesis $\mathbf{y}^*$. Then, the solution to the overdetermined system $\mathrm{R}_{\mathbf{x}} \cdot \breve{\mathbf{u}}^t = \mathbf{l}_{\mathbf{x}}$ is given by the equation:

$$\breve{\mathbf{u}}^t = (\mathrm{R}'_{\mathbf{x}} \cdot \mathrm{R}_{\mathbf{x}} + \beta \mathrm{I})^{-1} \, \mathrm{R}'_{\mathbf{x}} \cdot \mathbf{l}_{\mathbf{x}} \,, \tag{27}$$

with $\mathrm{R}_{\mathbf{x}} = \mathrm{G}^*_{\mathbf{x}} - \mathrm{G}_{\mathbf{x}}$ in this case.

4.4. Bayesian Predictive Adaptation

All three methods presented above rely on a single-best point estimation of the model parameters. In contrast, in Bayesian adaptation, the model parameters are considered to be random variables that have some kind of a priori distribution. Observing these random variables leads to a posterior density, which typically peaks at the optimal values of these parameters. Since the marginal likelihood where the model parameters have been marginalized is often intractable, in *Bayesian predictive adaptation* (BPA), this likelihood is approximated by sampling directly from the posterior distribution of the data given the model parameters. This leads to an approximation of the real distribution, rather than a point estimate, and usually entails more robust estimates. In this article, BPA is applied in an online scenario.

In BPA, Eq. 1 is reformulated as follows (see [18] for the full derivation):

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}} \int p(\mathbf{y}, \theta \mid \mathbf{x}; T, A) d\theta \approx \underset{\mathbf{y}}{\text{argmax}} \int p(\theta \mid T, A) p(\mathbf{y} \mid \mathbf{x}, \theta) d\theta \quad (28)$$

$$\approx \underset{\mathbf{y}}{\text{argmax}} \int p(A \mid \theta; T) p(\theta \mid T) p(\mathbf{y} \mid \mathbf{x}, \theta) d\theta, \quad (29)$$

where T represents the complete training set, A the adaptation data, and θ the model parameters. (i.e., either $\mathbf{h}$ or $\boldsymbol{\lambda}$). In the approximation in Eq. 28, several assumptions have been made. The first assumption is that output sentence $\mathbf{y}$ only depends on the model parameters and the current input sentence $\mathbf{x}$, and not on the complete training and adaptation data. The second one is that model parameters can be assumed to be independent of the current input sentence $\mathbf{x}$. From Eq. 28 to Eq. 29, the term $p(\theta \mid T, A)$ has been decomposed according to the Bayes theorem, where the normalization denominator can be neglected given that it has no influence on the maximization. This leads to a very intuitive decomposition: the last term, $p(\mathbf{y} \mid \mathbf{x}, \theta)$ is the same term that appears in Eq. 1; the middle term, $p(\theta \mid T)$ is a prior distribution over model parameters, which will account for rewarding those parameter sets that are similar to our prior knowledge; finally, $p(A \mid \theta; T)$ will account for biasing the final marginal likelihood towards A. Performing the integral over the complete parametric space forces the model to take into account all possible values of θ, thereby yielding more robust estimations of $p(\mathbf{y} \mid \mathbf{x})$.

Although computing the above integral is the correct thing to do from a theoretical point of view, in Bayesian learning, it is quite common to have intractable integrals. Hence, a random sampling $\mathcal{S}(\theta_T)$ is considered, with θ_T being the estimation of θ obtained at training time. In this work, this sampling will be performed by alternatively perturbing each one of the components of θ_T by a random amount, since such heuristic procedure is shown in [18] to perform well in SMT.

Considering A^t (i.e., the last A^t sentences validated by the user at time t) as the adaptation sample leads to an online variant of BPA, where

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}} \sum_{\forall\theta\in\mathcal{S}(\theta_T)} p(A^t \mid \theta; T)p(\theta \mid T)p(\mathbf{y} \mid \mathbf{x}_t, \theta). \tag{30}$$

Here, it has been assumed that θ_T remains invariable in time. Hence, $p(\theta \mid T)$ does not vary in time either and $\mathcal{S}(\theta_T)$ remains constant. Therefore, the online nature of Eq. 30 relies only on A^t, but it allows an efficient implementation, given that:

- $p(\theta \mid T)$ can be precomputed.
- $p(\mathbf{y}|\mathbf{x}, \theta)$ needs to be computed for every hypothesis and every test sentence.
- $p(A^t|\theta; T) = \prod_{\forall a\in A^t} p(\mathbf{x}_a, \mathbf{y}_a|\theta; T)$ only requires one division and one multiplication in order to incorporate the last sentence, since each one of the components within the product have already been computed when this component was the actual test sentence (see Figure 4 for the case of adapting $\boldsymbol{\lambda}$).

It is also shown in [18] that it is beneficial to consider a leveraging term α:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}} \sum_{\forall\theta\in\mathcal{S}(\theta_T)} \big(p(A^t \mid \theta; T)p(\mathbf{y} \mid \mathbf{x}_t, \theta)\big)^{\alpha} p(\theta \mid T), \tag{31}$$

whose role is similar to the α described in Section 3. Note that, in the case of BPA, there is no linear interpolation between the old parameters and the newly obtained ones since BPA does not compute a single point estimate of these parameters.

4.4.1. BPA for scaling factor adaptation

Plugging in the log-linear model in Eq. 3, assuming that $p(\theta|T) \sim \mathcal{N}(\theta_T, I{\cdot}\sigma)$, and considering as parameters the scaling factors $\boldsymbol{\lambda}$ of the log-linear model, yields:

$$p(\mathbf{y}|\mathbf{x}_t) \approx \sum_{\forall\boldsymbol{\lambda}\in\mathcal{S}(\boldsymbol{\lambda}_T)} \prod_{\forall a\in A^t} \frac{\exp \boldsymbol{\lambda}\cdot\mathbf{h}(\mathbf{x}_a, \mathbf{y}_a)}{\sum_{\mathbf{y}'}\exp \boldsymbol{\lambda}\cdot\mathbf{h}(\mathbf{x}_a, \mathbf{y}')} \mathcal{N}(\boldsymbol{\lambda}_T, I{\cdot}\sigma) \frac{\exp \boldsymbol{\lambda}\cdot\mathbf{h}(\mathbf{x}_t, \mathbf{y})}{\sum_{\mathbf{y}'}\exp \boldsymbol{\lambda}\cdot\mathbf{h}(\mathbf{x}_t, \mathbf{y}')}. \tag{32}$$

initialize $\mathcal{S}(\boldsymbol{\lambda}_T)$

for all $\boldsymbol{\lambda}_s \in \mathcal{S}(\boldsymbol{\lambda}_T)$ **do** $p(\boldsymbol{\lambda}_s) \leftarrow \mathcal{N}(\boldsymbol{\lambda}_s; \boldsymbol{\lambda}_T, \sigma)$ **end for**

while input sentences $\mathbf{x}_t$ in stream **do**

for all $\mathbf{y} \in nbest(\mathbf{x}_t)$; $\boldsymbol{\lambda}_s \in \mathcal{S}(\boldsymbol{\lambda}_T)$ **do**

$$B_{\mathbf{y},s} \leftarrow \frac{\exp \boldsymbol{\lambda}_s \cdot \mathbf{h}(\mathbf{x}_t, \mathbf{y})}{\sum_{\mathbf{y}' \in nbest(\mathbf{x}_t)} \exp \boldsymbol{\lambda}_s \cdot \mathbf{h}(\mathbf{x}_t, \mathbf{y}')}$$

end for

$$\hat{\mathbf{y}} \leftarrow \operatorname*{argmax}_{\mathbf{y}} \sum_{\boldsymbol{\lambda}_s \in \mathcal{S}(\boldsymbol{\lambda}_T)} \left(p(A^t \mid \boldsymbol{\lambda}) \cdot T_{\mathbf{y},s}\right)^{\alpha} \cdot p(\boldsymbol{\lambda}_s)$$

output $\hat{\mathbf{y}}$

for all $\boldsymbol{\lambda}_s \in \mathcal{S}(\boldsymbol{\lambda}_T)$ **do** $p(A^t \mid \boldsymbol{\lambda}) \leftarrow p(A^t \mid \boldsymbol{\lambda}) \cdot \frac{B_{\mathbf{y}^*,s}}{A^t_{1,s}}$ **end for**

$\text{dequeue}(A^t)$

$\text{enqueue}(A^t, B_{\mathbf{y}^*})$

end while

Figure 4: Pseudo-code for BPA online learning of the scaling factors $\boldsymbol{\lambda}$. B is a $N \times S$ matrix, where N is the size of $nbest(\mathbf{x})$ and S is the size of $\mathcal{S}(\boldsymbol{\lambda}_T)$. $B_{\mathbf{y}^*}$ is the row of B that corresponds to the best output hypothesis $\mathbf{y}^*$. In turn, A^t_a stands for term $B_{\mathbf{y}^*}$, but for adaptation sample a, which has been previously seen. Operation $\texttt{dequeue}(A^t)$ removes the first column of A^t (i.e., discards the oldest adaptation sample, A^t_1) and $\texttt{enqueue}(A^t, B_{\mathbf{y}^*})$ adds $B_{\mathbf{y}^*}$ as the last column of A^t.

Figure 4 shows the pseudo-code for adapting $\boldsymbol{\lambda}$ with BPA. A^t is treated as a $C \times S$ matrix, with C being the number of sentences seen before time t to be considered within BPA, and $S = |\mathcal{S}(\theta_T)|$.

4.4.2. BPA for feature function adaptation

Considering the feature function h_s as the model parameters (Section 3) yields

$$p(\mathbf{y} \mid \mathbf{x}_t) \approx \sum_{\forall \mathbf{u} \in \mathcal{S}(\mathbf{u}_T)} \prod_{\forall a \in A^t} \frac{\exp\left(h_r(\mathbf{x}_a, \mathbf{y}_a) + \mathbf{u} \cdot \mathbf{g}(\mathbf{x}_a, \mathbf{y}_a)\right)}{\sum_{\mathbf{y}'} \exp\left(h_r(\mathbf{x}_a, \mathbf{y}') + \mathbf{u} \cdot \mathbf{g}(\mathbf{x}_a, \mathbf{y}')\right)} \mathcal{N}(\mathbf{u}_T, I \cdot \sigma) \frac{\exp\left(h_r(\mathbf{x}_t, \mathbf{y}) + \mathbf{u} \cdot \mathbf{g}(\mathbf{x}_t, \mathbf{y})\right)}{\sum_{\mathbf{y}'} \exp\left(h_r(\mathbf{x}_t, \mathbf{y}) + \mathbf{u} \cdot \mathbf{g}(\mathbf{x}_t, \mathbf{y}')\right)} \,. \quad (33)$$

Although Eq. 33 formulates how to obtain an adequate feature function adaptation by means of BPA, implementing this formula is, in practice, too costly. For this reason, in the case of BPA, experiments are only conducted to adapt $\boldsymbol{\lambda}$.

5. Experiments

5.1. Experimental setup

Given that a true CAT scenario is very expensive for experimentation purposes since it requires a human translator to correct every hypothesis, we will simulate this scenario by using the reference sentences that are present in the test set. These sentences are fed one at a time, as would normally occur in an online CAT process. Even though the final TER scores are reported for the whole test set considered, each reference sentence was used for adaption only after the source sentence had been translated and its translation quality had been assessed. Hence, the translation quality reported corresponds to the average over the complete test set, even though the system had not been adapted at all for the first samples.

The most popular translation quality metrics are TER [23] and BLEU [24]. TER is an error metric that computes the minimum number of edits required to modify the system hypotheses so that they match the references. Possible edits include insertion, deletion, substitution of single words, and shifts of word sequences. BLEU is an accuracy metric that measures n-gram precision, with a penalty for sentences that are too short. For coherence, the quality metric used for assessing translation quality is also the one used for computing $\mathbf{y}^*$ (see Section 3). In this paper, we favor the use of TER, since BLEU implements a geometrical average which is zero whenever reference and hypothesis do not share a common 4-gram. Hence, BLEU is not appropriate for measuring translation quality at the

Table 1: Characteristics of the Europarl corpus and NC09 test set. OoV stands for "Out of Vocabulary" words, k stands for thousands of elements, and M stands for millions of elements.

		Spanish	English	French	English
Training	Sentences	1.3M		1.2M	
	Run. words	27.5M	26.6M	28.2M	25.6M
	Vocabulary	125.8k	82.6k	101.3k	81.0k
Development	Sentences	2000		2000	
	Run. words	60.6k	58.7k	67.3k	48.7k
	OoV. words	164	99	99	104

sentence level and $\mathbf{y}^*$ may not be appropriately defined.

As baseline system, we trained an SMT system on the Europarl training data, in the partition established in the Workshop on SMT of the ACL 2010[3]. Initially, the SMT system was trained using the training and development data provided that year. The Europarl corpus [32] is built from the transcription of European Parliament speeches published on the web. The data was collected by crawling the web between 1996 and 2010 in the 11 official languages of the European Union. Then it was aligned at the document level, split into sentences, normalized, tokenized, and aligned at the sentence level. This corpus has found a very widespread use in the SMT community, having been used for numerous SMT evaluation campaigns. We focus on the Spanish–English (Es–En) and French–English (Fr–En) language pairs. Corpus statistics are shown in Table 1.

The baseline system was built according to recent SMT evaluation campaigns [2, 33] by means of the open-source MT toolkit Moses [34], which was used in its default setup. In this setup, the trained SMT system features a statisti-

[3]http://www.statmt.org/wmt10/

cal log-linear model that includes a PB translation model, a language model, a distortion model, and word and phrase penalties. The PB translation model provides direct and inverse frequency-based and lexical-based probabilities for each phrase pair in the phrase-table. Phrase pairs are extracted from symmetrized word alignments generated by GIZA++ [35]. To model reordering, in addition to a negative-exponential on reordering distance, a model conditioned on phrases was estimated, namely the "orientation-bidirectional-fe" distortion model [36]. A 5-gram language model was estimated on the target side of the training data using Kneser-Ney smoothing [37] by means of SRILM [38]. The resulting 14 weights in Eq. 2 were estimated using MERT [39] on the Europarl development set.

Since our purpose is to analyze the performance of different online adaptation strategies and methods, in addition to Europarl, we also considered different test sets that do not belong to the parliamentary domain and that have been used in SMT evaluation campaigns, such as the News Commentary[4] (NC) 2009 and the TED[5] test sets. In addition, the learning rate α for every online adaptation algorithm was optimized on the NC 2008 test set. The News Commentary corpus was obtained from different news feeds and was used as test set for the 2010 ACL shared task on SMT [2]. For reasons of brevity, we only report the results with the NC 2009 test set and in English–Spanish translation, even though results involving other language pairs and test sets from other years were found to be consistent with the results presented here. The TED corpus is a compendium of public talks belonging to different domains, which was used as test set in the 2010 IWSLT shared task [33]. This corpus is only available for English–French translation and

[4]This corpus is available from http://www.statmt.org/wmt11/

[5]This corpus is available from http://iwslt2010.fbk.eu/

Table 2: Characteristics of News Commentary 2008, 2009, and TED test sets. OoV stands for "Out of Vocabulary" words with respect to the Europarl training set, k stands for thousands of elements, and M stands for millions of elements.

	NC08		NC09		TED	
	Spanish	English	Spanish	English	English	French
Sentences	2051		2525		1704	
Run. words	52.6k	49.9k	65.6k	68.1k	32.0k	33.9k
OoV. words	1029	958	1229	1358	278	692

was designed for translation *into* French. See Table 2 for NC and TED test set statistics. The performance of the different methods presented is compared on the NC09 set, and the TED test set is used to verify the conclusions drawn.

The aggressivity parameter C within PA was set to ∞ ($\frac{1}{C} = 0$ was used) following the work in [17]. The size of A^t in BPA was set to 100 and β was set to 0.01 for DRR, according to the preliminary experimentation carried out on other language pairs and other corpora. For PA, PCL, and BPA, instead of using the true best hypothesis, the best hypothesis within the N-best list was selected. The experiments that include the PA heuristic (see Section 4) are not reported since this heuristic did not bring significant improvements over the original PA algorithm.

5.2. Experimental results

As a first step, we analyzed the effect of varying the learning rate α in all studied online adaptation algorithms, as described in Eqs. (8), (6), and (31). The final TER score after processing all of the samples present in the NC08 test set is shown in Figure 5. Note that, in Figure 5, the scale of feature function adaptation is different to the one in scaling factor adaptation so that curves can be clearly distinguished. Interestingly, it was found that the optimum learning rates are rather

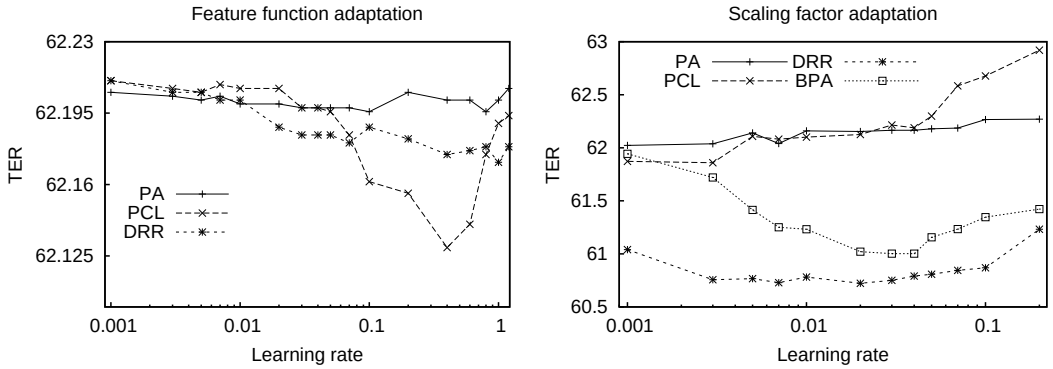

Figure 5: Influence of α on algorithm performance for NC08. N-best size was fixed to 500.

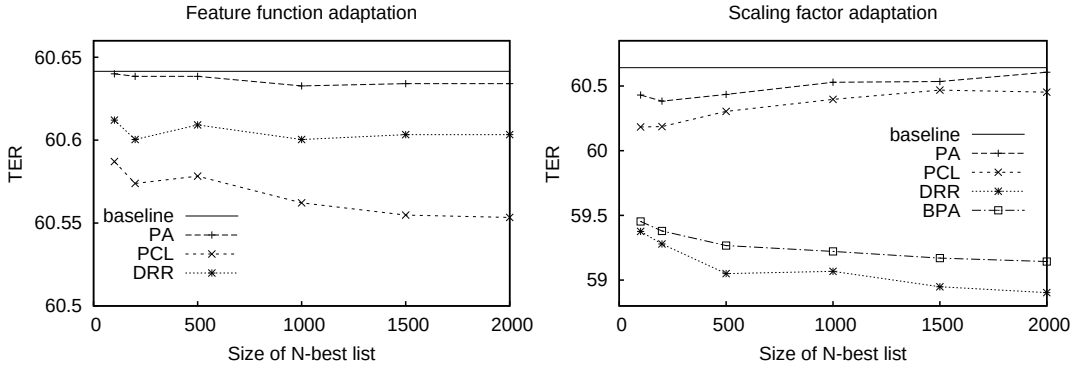

Figure 6: Final TER scores when adapting $\mathbf{h}$ or $\boldsymbol{\lambda}$ for NC09 test set in En$\rightarrow$Es direction.

consistent (~ 0.01) for all language pairs in the News Commentary test data.

Under the assumption that the quality of hypotheses is not necessarily correlated with their likelihood, larger N-best lists may include better hypotheses with a very low likelihood. Online predictors should be able to find these good hypotheses even if they appear deep in the N-best list. After setting the optimum α, the final translation quality obtained with varying sizes of $nbest(\mathbf{x})$ was measured on unseen test sets, i.e., the NC09 and TED test sets. The results of these

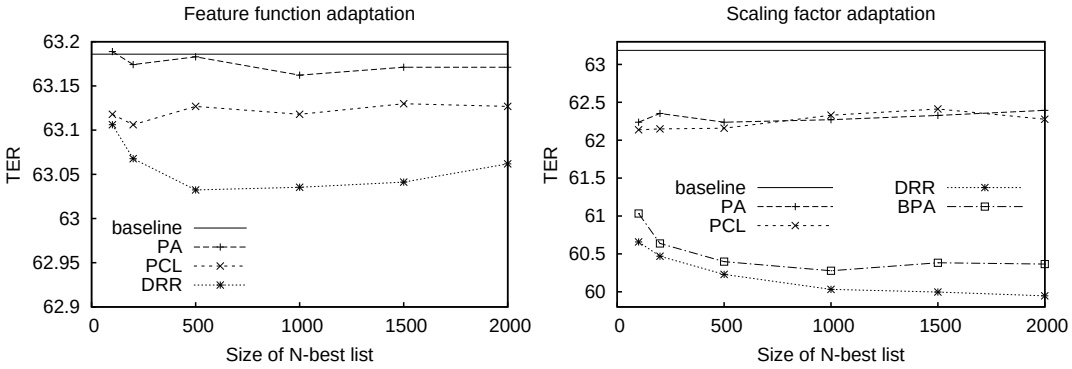

Figure 7: Final TER scores when adapting $\mathbf{h}$ or $\boldsymbol{\lambda}$ for the TED test set in En→Fr direction.

experiments are shown in Figure 6 for NC09 and in Figure 7 for TED.

When adapting $\boldsymbol{\lambda}$, the improvements obtained with DRR and BPA proved to be statistically significant at a 95% confidence level. Although the difference between BPA and DRR was not statistically significant, this difference was found to be consistent in all of the experiments. Moreover, it was found that DRR achieved a performance that was close to the maximum in a linear combination when the size of the N-best list was large enough (2000 hypotheses). Other differences, such as the one between PCL and PA were neither significant nor consistent.

When adapting $\mathbf{h}$, there was no algorithm that clearly outperformed the others. Differences were not found to be statistically significant, even though the methods studied did achieve consistent improvements in all of the experiments conducted.

All online learning algorithms performed better in scaling factor than in feature function adaptation, most probably due to the sparsity of the latter in comparison to the former: when adapting the scaling factors, only fourteen parameters needed to be adapted, versus around four million when adapting the feature functions $\mathbf{h}$. When adapting $\mathbf{h}$, there were as many parameters as bilingual phrases.

Table 3: Example of phrases that were used more than once when using the PCL algorithm. "Count" is the number of times that this phrase was used for translating the NC09 test set. The phrases that were seen more than 20 times mainly included punctuation marks and prepositions.

source phrase	target phrase	count
american	los	4
american	norteamericano	4
financial	financiera	2
financial	financieros	3
the financial crisis	la crisis financiera	9
the company	la empresa	20

To give a rough idea of the sparsity of the problem, it is worth mentioning the following: out of four million bilingual phrases, only $20,000$ were used to build the hypothesis used by the best system, around $3,500$ bilingual phrases were used more than once, and $1,500$ were used more than twice. Table 3 shows examples of bilingual phrases that were used more than once. The number of sentences affected by the feature function adaptation was also scarce. In $2,525$ sentences from the NC09 test set, 59 had an improvement in the TER score and 38 suffered a decrease in translation quality, as measured by the TER score when compared with the baseline. A positive example of the effect of promoting/demoting bilingual phrases when adapting $\mathbf{h}$ is shown in Figure 8 for PCL, which is the one that displays the best behavior in Figure 6. Those two sentences appear consecutively in the input stream. In the first sentence, "whip" is incorrectly translated by both the baseline and the PCL. However, the post-edited sentence used as reference indicates that the best translation for "whip" is "látigo". This bilingual phrase can be found in the phrase-table and PCL promotes it. The result can be observed in the next sentence, where PCL found the hypothesis with the right bilingual phrase.

`source`	sugar and whip for drivers	
`baseline`	el azúcar y oportunidades para los conductores	2
`PCL`	el azúcar y oportunidades para los conductores	2
`reference`	azúcar y látigo para los conductores	
`source`	the drivers are influenced by the sugar and whip system .	
`baseline`	los conductores están influidos por el azúcar y la disciplina del sistema .	9
`PCL`	los conductores están influidos por el azúcar y látigo .	5
`reference`	para los conductores vale el sistema de azúcar y látigo .	

Figure 8: Translation examples from NC09 when using PCL to adapt feature functions. `source` stands for input $\mathbf{x}$, `baseline` corresponds to the output of the non-adaptive system, `PCL` is the technique described in Sec. 4.2, and `reference` is the user post-edited translation. PCL found a more appropriate translation after promoting the bilingual phrase (`whip`,`látigo`) observed previously. The number of editions required to post-edit the hypotheses is shown in the last column.

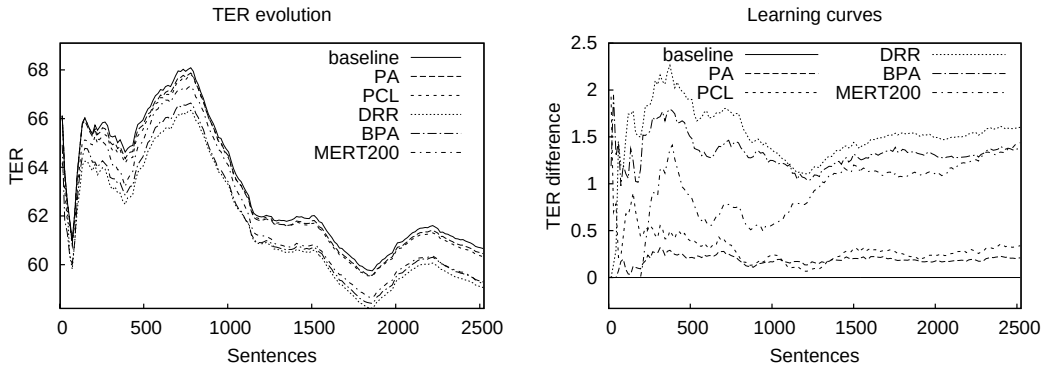

Figure 9: TER evolution and learning curves when adapting $\boldsymbol{\lambda}$ within the NC09 test set. Only 1 every 15 points has been drawn so that the plots are clearly distinguishable.

It can be observed that BPA and DRR tended to obtain better results when the size of the N-best list was increased. We consider this to be important, since it means that, when these algorithms are provided with more information, they are able to deal with it properly, without tending to yield over-trained estimations.

The evolution of the different online adaptation algorithms throughout the whole NC09 test set is shown in Figure 9. In this plot, the size of the N-best list was set to 500. The plot on the left shows the TER score averaged up to the t-th sentence considered, since plotting individual sentence scores would result in a very chaotic, unreadable plot given that the differences in translation quality between two single sentences may be very large; in fact, this chaotic behavior can still be seen in the first 100 sentences. The plot on the right shows the difference in translation quality between the online learning algorithms and the baseline. For comparison purposes, both plots also display an additional curve named `MERT200`, which is the result of performing a full re-estimation of $\boldsymbol{\lambda}$ by optimizing TER on the previous 200 sentences seen using the Z-MERT toolkit [40]. For computational reasons, $\boldsymbol{\lambda}$ was only re-estimated every 200 sentences. We analyzed the effect of re-estimating $\boldsymbol{\lambda}$ every 200 sentences on all the data seen up to that point, but the result is omitted here because this strategy accumulated many errors between sentences 200 and 800 and resulted in 4 TER points worse than the baseline. Although the learning curves peaked at about 750 sentences, this was not consistent throughout all experiments since this peak ranged from 300 to 1500 in other language pairs. Since the particular shape of the learning curves depends strongly on the test set chosen, the only information that can be extracted is whether or not the implemented algorithms provide improvements.

To gain some insight about what happens during the adaptation of $\boldsymbol{\lambda}$, different statistics computed after processing the whole NC09 set are shown in Table 4. Note that DRR, BPA, and $\mathbf{y}^*$ try to minimize TER, which does not explicitly take into account sentence length, as in the case of BLEU. This can also be observed when looking at the BLEU scores: both BPA and DRR are severely penalized by

Table 4: Different statistics obtained from the online learning methods. $|\mathbf{y}|$ stands for average sentence length (27.0 for the references). Brev. pen. is the brevity penalty within BLEU.

setup	$\lvert\mathbf{y}\rvert$	BLEU	n-gram precision	brev. pen.
baseline	26.6	22.0	57.9/28.3/16.0/9.4	0.985
DRR	25.1	21.4	59.5/29.2/16.6/9.8	0.929
BPA	25.6	21.7	59.1/29.0/16.4/9.7	0.950
$\mathbf{y}^*$	26.2	26.6	62.6/34.1/20.7/12.9	0.968

the brevity penalty, leading to slightly lower BLEU scores than the baseline. Since n-gram precision is notably higher, we understand that improvements achieved in TER are due to a better lexical choice of the phrases involved.

One last consideration involves computation time. When adapting $\boldsymbol{\lambda}$, the implemented procedures take about 100 seconds to re-rank the complete test set (90 minutes for `MERT200`), whereas in the case of adapting $\mathbf{h}$ the time consumed is about 25 minutes by a single-threaded implementation in an Intel Core 2 Quad CPU at 2.66GHz. We consider this to be important, since, in a CAT scenario, the user is actively waiting for the system to produce a hypothesis.

6. Conclusions and future work

Two important aspects of pattern recognition have been carefully studied in their instantiation to machine translation. The first one consists in finding the best possible representation of the observations (sentences) that leads to different adaptation strategies. The second one involves the study of the appropriateness of several online learning algorithms to adjust the prediction mechanisms after every sample is presented to the system. Thus, four online learning algorithms were used on a sentence-by-sentence basis for feature function and scaling factor

adaptation, achieving a different level of success. When these algorithms were applied to feature function adaptation, improvements achieved were not consistent. One possible reason was that the amount of corrective information provided by the user is relatively small when compared to the number of feature function parameters. In scaling factor adaptation, both discriminative ridge regression and Bayesian predictive adaptation provided significant positive results, and translation quality increased with the size of the N-best list. In our opinion, this is a desirable behavior since it implies that additional information has a positive effect on the performance of the applied algorithm. Based on this evidence, we intend to implement Bayesian predictive adaptation and discriminative ridge regression as applied to scaling factor adaptation into the decoder itself in the hope of achieving even greater improvements. We also plan to study the effect of combining both feature function and scaling factor adaptation.

Acknowledgments

This paper is based upon work supported by the EC (FEDER/FSE) and the Spanish MICINN under projects MIPRCV "Consolider Ingenio 2010" (CSD2007-00018) and iTrans2 (TIN2009-14511). This work is also supported by the Spanish MITyC under the erudito.com (TSI-020110-2009-439) project, by the Generalitat Valenciana under grant Prometeo/2009/014, and by the UPV under grant 20091027. The authors would like to thank the anonymous reviewers for their useful and constructive comments.

References

[1] H. Christensen, Speaker Adaptation of Hidden Markov Models using Maximum Likelihood Linear Regression, Ph.D. thesis, Aalborg University, Den-

mark, 1996.

[2] C. Callison-Burch, et al., Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation, in: Proc. of the Joint 5th workshop on SMT and MetricsMATR, pp. 17–53.

[3] A. Blum, On-line algorithms in machine learning, in: Proc. of the workshop on On-Line Algorithms, Dagstuhl, 1996, pp. 306–325.

[4] Y. Zhang, M. S. Scordilis, Effective online unsupervised adaptation of gaussian mixture models and its application to speech classification, Pattern Recognition Letters 29 (2008) 735 – 744.

[5] Q. Huo, C.-H. Lee, On-line adaptive learning of the continuous density hidden markov model based on approximate recursive bayes estimate, IEEE Transactions on Speech and Audio Processing 5 (1997) 161 –172.

[6] K.-C. Lee, D. Kriegman, Online learning of probabilistic appearance manifolds for video-based recognition and tracking, in: Proc. of IEEE intl. conf. on Computer Vision and Pattern Recognition, 2005, pp. 852–859.

[7] C. Callison-Burch, C. Bannard, J. Schroeder, Improving statistical translation through editing, in: Proc. of 9th EAMT workshop "Broadening horizons of machine translation and its applications", 2004, pp. 26–32.

[8] S. Barrachina et al., Statistical approaches to computer-assisted translation, Computational Linguistics 35 (2009) 3–28.

[9] P. Koehn, Statistical Machine Translation, Cambridge University Press, 2010.

[10] F. J. Och, H. Ney, Discriminative training and maximum entropy models for statistical machine translation, in: Proc. of the 40th annual conf. of the Association for Computational Linguistics, 2002, pp. 295–302.

[11] J. L. Gauvain, C. H. Lee, Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains, IEEE Transactions on Speech and Audio Processing 2 (1994) 291–298.

[12] H. Daumé III, Frustratingly easy domain adaptation, in: Proc. of the 45th annual conf. of the Association for Computational Linguistics, 2007, pp. 256–263.

[13] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society 39 (1977) 1–38.

[14] D. Ortiz-Martínez, I. García-Varea, F. Casacuberta, Online learning for interactive statistical machine translation, in: Proc. of the annual conf. of the North American chapter of the Association for Computational Linguistics, 2010, pp. 546–554.

[15] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, Psychological Review 65 (1958) 386–408.

[16] K. Crammer, et al., Online passive-aggressive algorithms, Journal of Machine Learning Research 7 (2006) 551–585.

[17] G. Reverberi, et al., Online learning algorithms for computer-assisted translation, Deliverable D4.2, SMART: Statistical Multilingual Analysis for Retrieval and Translation (2008).

[18] G. Sanchis-Trilles, F. Casacuberta, Log-linear weight optimisation via bayesian adaptation in statistical machine translation, in: Proc. of the intl. conf. on Computational Linguistics, 2010, pp. 1077–1085.

[19] P. Liang, A. Bouchard-Côté, D. Klein, B. Taskar, An end-to-end discriminative approach to machine translation, in: Proc. of the 44th Annual Meeting of the Association for Computational Linguistics, 2006, pp. 761–768.

[20] T. Watanabe, J. Suzuki, H. Tsukada, H. Isozaki, Online large-margin training for statistical machine translation, in: Proc. of the conf. on Empirical Methods in Natural Language Processing, 2007, pp. 764–773.

[21] D. Chiang, Y. Marton, P. Resnik, Online large-margin training of syntactic and structural translation features, in: Proc. of the conf. on Empirical Methods in Natural Language Processing, 2008, pp. 224–233.

[22] A. Arun, P. Koehn, Online learning methods for discriminative training of phrase based statistical machine translation, in: Proc. of the 11th Machine Translation Summit, 2007, pp. 15–20.

[23] M. Snover, other, A study of translation edit rate with targeted human annotation, in: Proc. of the 7th biennial conf. of the Association for Machine Translation in the Americas, 2006, pp. 223–231.

[24] K. Papineni, S. Roukos, T. Ward, W. J. Zhu, Bleu: A method for automatic evaluation of machine translation, in: Proc. of the 40th annual conf. of the Association for Computational Linguistics, 2002, pp. 311–318.

[25] C. Stauffer, E. Grimson, Learning patterns of activity using real-time tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000) 747–757.

[26] P. Martínez-Gómez, G. Sanchis-Trilles, F. Casacuberta, Passive-aggressive for on-line learning in statistical machine translation, in: Proc. of the Iberian conf. on Pattern Recognition and Image Analysis, 2011, pp. 240–247.

[27] M. Collins, Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms, in: Proc. of the conf. on Empirical Methods in Natural Language Processing, 2002, pp. 1–8.

[28] C. España-Bonet, L. Màrquez, Robust estimation of feature weights in statistical machine translation, in: Proc. 14th annual conf. of the European Association for Machine Translation, 2010.

[29] S. Kung, J. Taur, Decision-based neural networks with signal/image classification applications, IEEE Transactions on Neural Networks 6 (1995) 170 –181.

[30] R. D. Brown, Taming structured perceptrons on wild feature vectors, in: Proc. of the joint 5th workshop on SMT and MetricsMATR, 2010, pp. 384–391.

[31] P. Martínez-Gómez, G. Sanchis-Trilles, F. Casacuberta, Online learning via dynamic reranking for computer assisted translation, in: Proc. of the 12th intl. conf. on Computational Linguistics and Intelligent Text Processing - Volume II, 2011, pp. 93–105.

[32] P. Koehn, Europarl: A parallel corpus for statistical machine translation, in: Proc. of the 10th Machine Translation Summit, 2005, pp. 79–86.

[33] M. Paul, M. Federico, S. Stüker, Overview of the IWSLT 2010 evaluation campaign, in: Proc. of the 2010 intl. workshop on Spoken Language Translation, 2011, pp. 3–27.

[34] P. Koehn et al., Moses: Open source toolkit for statistical machine translation, in: Proc. of the ACL Demo and Poster Sessions, 2007, pp. 177–180.

[35] F. J. Och, H. Ney, A systematic comparison of various statistical alignment models, Computational Linguistics 29 (2003) 19–51.

[36] P. Koehn, et al., Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation, in: Proc. of the intl. workshop on Spoken Language Translation, 2005.

[37] S. F. Chen, J. Goodman, An empirical study of smoothing techniques for language modeling, Computer Speech and Language 4 (1999) 359–393.

[38] A. Stolcke, SRILM – an extensible language modeling toolkit, in: Proc. of the 7th intl. conf. on Spoken Language Processing, 2002, pp. 901–904.

[39] F. J. Och, Minimum error rate training for statistical machine translation, in: Proc. of the 41st annual conf. of the Association for Computational Linguistics, 2003, pp. 160–167.

[40] O. F. Zaidan, Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems, The Prague Bulletin of Mathematical Linguistics 91 (2009) 79–88.