D4.2: Progress Report on Adaptive Translation Models

Daniel Ortiz-Martínez, Germán Sanchis-Trilles, Jesús González-Rubio, Francisco Casacuberta

Distribution: Public

CASMACAT
Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation
ICT Project 287576 Deliverable D4.2

Project funded by the European Community under the Seventh Framework Programme for Research and Technological Development.
<table>
<thead>
<tr>
<th>Project ref no.</th>
<th>ICT-287576</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym</td>
<td>CASMACAT</td>
</tr>
<tr>
<td>Project full title</td>
<td>Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation</td>
</tr>
<tr>
<td>Instrument</td>
<td>STREP</td>
</tr>
<tr>
<td>Thematic Priority</td>
<td>ICT-2011.4.2 Language Technologies</td>
</tr>
<tr>
<td>Start date / duration</td>
<td>01 November 2011 / 36 Months</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contractual date of delivery</td>
<td>October 31, 2013</td>
</tr>
<tr>
<td>Actual date of delivery</td>
<td>October 29, 2013</td>
</tr>
<tr>
<td>Date of last update</td>
<td>D4.2</td>
</tr>
<tr>
<td>Deliverable number</td>
<td>D4.2</td>
</tr>
<tr>
<td>Deliverable title</td>
<td>Progress Report on Adaptive Translation Models</td>
</tr>
<tr>
<td>Type</td>
<td>Report</td>
</tr>
<tr>
<td>Status &amp; version</td>
<td>Revised after first evaluation meeting</td>
</tr>
<tr>
<td>Number of pages</td>
<td>39</td>
</tr>
<tr>
<td>Contributing WP(s)</td>
<td>WP4</td>
</tr>
<tr>
<td>WP / Task responsible</td>
<td>UPVLC</td>
</tr>
<tr>
<td>Other contributors</td>
<td>CS</td>
</tr>
<tr>
<td>Internal reviewer</td>
<td>Philipp Koehn</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Daniel Ortiz-Martínez, Germán Sanchis-Trilles, Jesús González-Rubio, Francisco Casacuberta</td>
</tr>
<tr>
<td>EC project officer</td>
<td>Kimo Rossi</td>
</tr>
<tr>
<td>Keywords</td>
<td></td>
</tr>
</tbody>
</table>

The partners in CASMACAT are:
University of Edinburgh (UEDIN)
Copenhagen Business School (CBS)
Universitat Politècnica de València (UPVLC)
Celer Soluciones (CS)

For copies of reports, updates on project activities and other CASMACAT related information, contact:

The CASMACAT Project Co-ordinator
Philipp Koehn, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB, United Kingdom
pkoehn@inf.ed.ac.uk
Phone +44 (131) 650-8287 - Fax +44 (131) 650-6626

Copies of reports and other material can also be accessed via the project’s homepage:
http://www.casmacat.eu/

© 2012, The Individual Authors
No part of this document may be reproduced or transmitted in any form, or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission from the copyright owner.
Executive Summary

This document contains information about the progress made on WP4 during the second year of the project. The main objective of this WP is to deal with adaptation in interactive translation prediction (ITP): algorithms, sentence selection and domain and user adaptation. The specific objectives are: (1) To develop algorithms for an efficient model adaptation; (2) To develop technique for efficient sentence selection for adaptation; (3) To explore approaches for an effective adaptation to new domains and users.

The work developed during the first year of the project affected the three different tasks contained in the work package, namely Task 4.1, Task 4.2 and Task 4.3:

• **Task 4.1: On-line Learning for Interactive Translation Prediction** (month 1-24)
  The work carried out during this second year of the project has been devoted to clarify some important properties of online learning when applied to SMT. For this purpose, the previous experimentation that was presented in the Deliverable 4.1 has been extended. Such a previous experimentation was focused on demonstrating the feasibility of the application of online learning to the SMT framework. However, some crucial aspects of online learning were not studied, such as its performance with respect to a conventional, batch learning algorithm, or the impact of update frequency in the system performance. The new experiments allow us to clarify such aspects. In addition to this, previous experiments were executed on corpora of a small or medium size, mainly due to the computational requirements of processing larger corpora. A significant part of the work carried out within Task 4.1 has been focused on creating the software tools that are necessary to process large corpora as well as the integration of such tools in the Casmacat workbench.

  Additionally, we have also extended the work concerning the online learning of the log-linear weights. The main problem observed during the first year of the period allotted for this task was that the improvements achieved by different techniques as applied in a conventional SMT setup did not carry on to the IMT framework, and sometimes even introduced losses in performance (see D4.1 for more details). In order to cope for this problem, the different algorithms for online learning of the weights were redefined, and some encouraging results were observed by obtaining weight samples from a Gaussian distribution and hence yielding different wordgraphs.

• **Task 4.2: Active Learning in Interactive Translation Prediction** (month 13-24)
  A cost-sensitive active learning (AL) framework for interactive translation prediction (ITP) has been proposed. Such an AL framework tries to make the translation process as efficient as possible by maximising the obtained translation quality per unit of user supervision effort. The proposed cost-sensitive AL framework boosts the productivity of existing ITP technology by addressing its two potential drawbacks. On the one hand, we do not require the user to exhaustively supervise all translations. Instead, we propose a selective interaction protocol where the user only supervises a subset of informative translations. This selective supervision is similar to the active interaction protocol studied in Task 2.3 in WP2, but applied at the sentence-level. On the other hand, the batch SMT model typically used by ITP systems is replaced by an incremental SMT model that utilises user feedback to continually update its parameters after deployment.

• **Task 4.3: Domain and User Adaptation** (month 13-30)
  The adaptation problem arises when two very different sets of training data are available, yielding two different sets of model parameters, namely, the training data (obtained from a general domain) and the adaptation data (obtained from a specific domain of interest). Then, the challenge is to modify the SMT system appropriately by taking into consideration both data sets. This definition of adaptation is specially appropriate for the Bayesian
**learning paradigm**, where the model parameters are treated as (hidden) random variables governed by some kind of a priori distribution. In the work conducted so far, we focus on the Bayesian adaptation of the weights of the log-linear combination of features present in every state-of-the-art SMT system. Even though these weights are not very numerous, providing the system with appropriate estimates for these weights is critical.

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Online Learning for Interactive Translation Prediction</td>
<td>5</td>
</tr>
<tr>
<td>1.1</td>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>1.2</td>
<td>Extended Online Learning Experimentation</td>
<td>5</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Log-Linear SMT Model with Online Learning</td>
<td>5</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Corpus</td>
<td>7</td>
</tr>
<tr>
<td>1.2.3</td>
<td>Evaluation Measures</td>
<td>7</td>
</tr>
<tr>
<td>1.2.4</td>
<td>EM Algorithm Convergence Experiments</td>
<td>8</td>
</tr>
<tr>
<td>1.2.5</td>
<td>Impact of Update Frequency</td>
<td>9</td>
</tr>
<tr>
<td>1.2.6</td>
<td>Batch versus Online Learning</td>
<td>9</td>
</tr>
<tr>
<td>1.2.7</td>
<td>Conclusions</td>
<td>10</td>
</tr>
<tr>
<td>1.3</td>
<td>Distributed Implementation of Phrase-Based Model Estimation</td>
<td>11</td>
</tr>
<tr>
<td>1.4</td>
<td>Online Learning of the Log-Linear Weights</td>
<td>11</td>
</tr>
<tr>
<td>1.4.1</td>
<td>Introduction</td>
<td>11</td>
</tr>
<tr>
<td>1.4.2</td>
<td>Development</td>
<td>11</td>
</tr>
<tr>
<td>1.4.3</td>
<td>Conclusions</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>Active Learning for Interactive Translation Prediction</td>
<td>12</td>
</tr>
<tr>
<td>2.1</td>
<td>Introduction</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>Development</td>
<td>13</td>
</tr>
<tr>
<td>2.3</td>
<td>Conclusions</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>Domain and User Adaptation</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>Development</td>
<td>14</td>
</tr>
<tr>
<td>3.3</td>
<td>Markov chain Monte Carlo</td>
<td>16</td>
</tr>
<tr>
<td>3.4</td>
<td>Experiments</td>
<td>17</td>
</tr>
<tr>
<td>3.5</td>
<td>Corpora</td>
<td>18</td>
</tr>
<tr>
<td>3.6</td>
<td>Comparison between BPA and parameter re-estimation</td>
<td>19</td>
</tr>
<tr>
<td>3.7</td>
<td>Conclusions</td>
<td>21</td>
</tr>
<tr>
<td>3.8</td>
<td>Future work</td>
<td>22</td>
</tr>
</tbody>
</table>

Bibliography | 23 |

Attachment A | 27 |
1 Online Learning for Interactive Translation Prediction

1.1 Introduction

During the first year of the project, we developed previous work on online learning estimation for statistical machine translation (SMT). Such an extension included a detailed mathematical derivation of the statistical models involved in the translation process, as well as experiments with a new corpus extracted from the Bulletin of the European Union (the EU corpus).

The online learning experiments that have been presented so far, were focused on demonstrating the utility and feasibility of online learning in an SMT scenario. In particular, it was demonstrated that online learning for SMT can be used to incrementally update the parameters of the SMT model in real time. Such updates are effective when the system has to learn from scratch or from previously estimated models. However, some crucial aspects of online learning were not studied, such as its performance with respect to a conventional, batch learning algorithm, or the impact of update frequency in the system performance.

Additionally, the previously reported experiments were obtained using small or medium size corpora. One reason for not extending the experiments to larger corpora was the high computational requirements of model estimation. A significant part of the work carried out within Task 4.1 during this year has been focused on preparing the software tools that were necessary to overcome the difficulties that arise when handling large corpora, as well as the integration of such tools within the CASMACAT workbench.

In the following sections, we present the additional online learning experiments that were carried out during this second year of the project (Section 1.2), as well as a brief explanation of the implementation work that was necessary to be able to execute experiments on large corpora (Section 1.3). Finally, some concluding remarks are added (Section 1.2.7).

1.2 Extended Online Learning Experimentation

1.2.1 Log-Linear SMT Model with Online Learning

Here we briefly introduce the log-linear model with online learning capabilities that was defined in Deliverable 4.1 (first year deliverable for WP4). The statistical approach to MT formalises the problem of generating translations under a statistical point of view. Given a sentence, \( x \equiv x^i_1 \equiv x_1 \cdots x_j \cdots x_J \) in the source language \( X \), we want to find its equivalent target sentence \( y \equiv y^i_1 \equiv y_1 \cdots y_i \cdots y_I \) in the target language \( Y \). The general formulation of the log-linear approach for SMT is as follows:

\[
\hat{y}_1^i = \arg \max_{I, y_1^i} \left\{ \sum_{m=1}^M \lambda_m h_m(x_1^i, y_1^i) \right\}
\]

(1)

Current log-linear models for SMT are strongly focused on the use of the so-called phrase-based models \[22\]. The basic idea of phrase-based translation is to segment the source sentence into phrases, then to translate each source phrase into a target phrase, and finally to reorder the translated target phrases in order to compose the target sentence. The decisions made during the translation process can be summarised by means of a hidden alignment variable: \( \hat{a}_1^j \). It is common to modify expression given by Equation 1 to reflect the important role of phrase-based models in the translation process. For this purpose, the hidden alignment variable

---

\( x_j \) and \( y_i \) note the \( i \)’th word and the \( j \)’th word of the sentences \( x^i_1 \) and \( y_1 \) respectively.
\( \hat{a}_t^K \) is introduced. Since there are multiple phrase alignments between the source and the target sentences, we would be interested in that of maximum score:

\[
\hat{y}_1^I = \arg \max_{I, \hat{y}_1^I} \left\{ \max_{\hat{a}_1^K} \sum_{m=1}^M \lambda_m h_m(x_1^I, y_1^I, \hat{a}_1^K) \right\}
\]

(2)

According to the previous equation, we introduce a set of seven feature functions in our log-linear model (from \( h_1 \) to \( h_7 \)): an \( n \)-gram language model (\( h_1 \)), a source sentence-length model (\( h_2 \)), inverse and direct phrase-based models (\( h_3 \) and \( h_4 \) respectively), a target phrase-length model (\( h_5 \)), a source phrase-length model (\( h_6 \)), and a distortion model (\( h_7 \)).

- **\( n \)-gram language model (\( h_1 \))**
  \[ h_1(y_1^I) = \log(\prod_{i=1}^{I-1} p(y_i|y_{i-1}^{i-1})) \]
  where \( p(y_i|y_{i-1}^{i-1}) \) is defined as follows:

  \[
P(y_i|y_{i-1}^{i-1}) = \frac{\max\{c_X(y_{i-1}^{i-1}) - D_n, 0\}}{c_X(y_{i-1}^{i-1})} + \frac{D_n}{c_X(y_{i-1}^{i-1}) \cdot N_{i+1}(y_{i-1}^{i-1}) \cdot p(y_i|y_{i-1}^{i-1})}
  \]
  \[
  \text{where } D_n = \frac{c_{n,1}}{c_{n,1} + 2c_{n,2}} \text{ is a fixed discount (} c_{n,1} \text{ and } c_{n,2} \text{ are the number of } n \text{-grams with one and two counts respectively), } N_{i+1}(y_{i-1}^{i-1}) \cdot \text{is the number of unique words that follows the history } y_{i-1}^{i-1} \text{ and } c_X(y_{i-1}^{i-1}) \text{ is the count of the } n \text{-gram } y_{i-1}^{i-1}, \text{ where } c_X(\cdot) \text{ can represent true counts } c_T(\cdot) \text{ or modified counts } c_M(\cdot). \text{ True counts are used for the higher order } n \text{-grams and modified counts for the lower order } n \text{-grams. Given a certain } n \text{-gram, its modified count consists in the number of different words that precede this } n \text{-gram in the training corpus.}
\]

Equation (3) corresponds to the probability given by an \( n \)-gram language model with an interpolated version of the Kneser-Ney smoothing \( [3] \).

- **source sentence-length model (\( h_2 \))**
  \[ h_2(x_1^I, y_1^I) = \log(p(J|I)) = \log(\phi_I(J+0.5) - \phi_I(J-0.5)) \]
  where \( \phi_I(\cdot) \) denotes the cumulative distribution function (cdf) for the normal distribution (the cdf is used here to integrate the normal density function over an interval of length 1). We use a specific normal distribution with mean \( \mu_I \) and standard deviation \( \sigma_I \) for each possible target sentence length \( I \).

- **inverse and direct phrase-based models (\( h_3, h_4 \))**
  \[ h_3(x_1^I, y_1^I, \hat{a}_1^K) = \log(\prod_{k=1}^K p(\bar{x}_k|\bar{y}_a_k)) \]
  where \( p(\bar{x}_k|\bar{y}_a_k) \) is defined as follows:

  \[
p(\bar{x}_k|\bar{y}_a_k) = \beta \cdot p_{\text{phr}}(\bar{x}_k|\bar{y}_a_k) + (1 - \beta) p_{\text{hmm}}(\bar{x}_k|\bar{y}_a_k)
  \]
  \[
  \text{In Equation (4), } p_{\text{phr}}(\bar{x}_k|\bar{y}_a_k) \text{ denotes the probability given by a statistical phrase-based dictionary used in regular phrase-based models (see [22] for more details). } p_{\text{hmm}}(\bar{x}_k|\bar{y}_a_k) \text{ is the probability given by an HMM-based (intra-phrase) alignment model (see [40]):}
\]

  \[
p_{\text{hmm}}(\bar{x}|\bar{y}) = \epsilon \sum_{a^{(a)}_1 \cdot \ldots \cdot \bar{y}} \prod_{j=1}^{\bar{y}} p(\bar{x}_j|\bar{y}_a_j) \cdot p(a_j|a_{j-1}, |\bar{y}|)
  \]
  \[
  \text{The HMM-based alignment model probability is used here for smoothing purposes.}
\]

Analogously \( h_3 \) is defined as:

\[ h_3(x_1^I, y_1^I, \hat{a}_1^K) = \log(\prod_{k=1}^K p(\bar{y}_a_k|\bar{x}_k)) \]

(4)

\footnote{\( I \) is the length of \( y_1^I \), \( y_0 \) denotes the begin-of-sentence symbol, \( y_{|y|+1} \) denotes the end-of-sentence symbol, \( y_i^I \equiv y_{i-1} \ldots y_j \)}
Table 1: English–Spanish Europarl corpus statistics.

<table>
<thead>
<tr>
<th></th>
<th>English</th>
<th>Spanish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>1 547 596</td>
<td>3 411 6128</td>
</tr>
<tr>
<td>Running</td>
<td>33 124 966</td>
<td>34 116 128</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>96 741</td>
<td>146 288</td>
</tr>
<tr>
<td>Dev</td>
<td>3003</td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td>72 988</td>
<td>78 888</td>
</tr>
<tr>
<td>Running</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td>64 809</td>
<td>70 562</td>
</tr>
<tr>
<td>Running</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **target phrase-length model** (h₅)
  \[ h₅(x₁, y₁, ˆa^K) = \log(\prod_{k=1}^{K} p(\hat{y}_k)), \]  where \( p(\hat{y}_k) = \delta(1-\delta)|\hat{y}_k| \). h₅ implements a target phrase-length model by means of a geometric distribution with probability of success on each trial δ.

The use of a geometric distribution penalises the length of target phrases.

- **source phrase-length model** (h₆)
  \[ h₆(x₁, y₁, ˆa^K) = \log(\prod_{k=1}^{K} p(\hat{x}_k | \hat{y}_k)), \]
  where \( p(|\hat{x}_k| | \hat{y}_k) = \frac{1}{1+\tau} \delta(1-\delta)\text{abs}(|\hat{x}_k|-|\hat{y}_k|), \tau = \sum_{i=1}^{N} |\hat{y}_k|-1 \delta(1-\delta)^i \) and \( \text{abs}(\cdot) \) is the absolute value function. A geometric distribution (with scaling factor \( \frac{1}{1+\tau} \)) is used to model this feature (it penalises the difference between the source and target phrase lengths).

- **distortion model** (h₇)
  \[ h₇(\hat{a^K}) = \log(\prod_{k=1}^{K} p(\hat{a}_k | \hat{a}_{k-1})), \]  where \( p(\hat{a}_k | \hat{a}_{k-1}) = \frac{1}{2^\tau} \delta(1-\delta)\text{abs}(\hat{a}_k-\hat{a}_{k-1}), \) \( \hat{a}_k \) denotes the beginning position of the source phrase covered by \( \hat{a}_{k-1} \), \( l_{\hat{a}_{k-1}} \) denotes the last position of the source phrase covered by \( \hat{a}_{k-1} \). A geometric distribution (with scaling factor \( \frac{1}{2^\tau} \)) is used to model this feature (it penalises the reordering).

After defining the different feature functions that compose the log-linear model, it is possible to define a set of incremental update rules that allows us to incorporate the knowledge provided by the user (the details of the update rules can be viewed in the previous year deliverable, D4.1).

### 1.2.2 Corpus

The experiments were performed using the English–Spanish language pair of the Europarl corpus [21]. The Europarl corpus is extracted from the proceedings of the European Parliament, which are written in the different languages of the European Union. In our experiments we have used the version that was created for the shared task of the ACL 2013 Workshop on Statistical Machine Translation [6]. To simplify the experiments, all those sentences whose length in words were greater than 40 were removed from the training set. Regarding the language pairs under consideration, again, we will translate from the English to Spanish. Table 1 shows the main figures of training, development and test sets. The Europarl corpus constitutes one good example of a complex, real-world translation task that is also very well-known in the MT scientific community.

### 1.2.3 Evaluation Measures

Since we want to evaluate the performance of our proposed SMT system in an IMT scenario, we need to estimate the effort required from the user to produce correct translations using the system. To this end, we use the target references to simulate the translations that the user has in mind. The first translation hypothesis for each given source sentence is compared with a
single reference translation and the longest common character prefix (LCP) is obtained. The first non-matching character is replaced by the corresponding reference character and then a new system translation is produced. This process is iterated until a full match with the reference is obtained. Each computation of the LCP would correspond to the user looking for the next error and moving the pointer to the corresponding position of the translation hypothesis. We refer to a pointer movement as a *mouse-action*. On the other hand, each character replacement would correspond to a *key-stroke* of the user. If the first non-matching character is the first character of the new system hypothesis in a given interaction, no LCP computation is needed; that is, no pointer movement would be made by the user. Bearing this in mind, we define the following IMT evaluation measure:

- **Key-stroke and mouse-action ratio** (KSMR): KSMR [4] is the number of key-strokes plus number of mouse-actions divided by the total number of reference characters.

It is worthy of note that KSMR assumes that both key-strokes and mouse-actions requires the same effort from the user. This constitutes an approximation, since these two actions are different and require different types of effort [30].

### 1.2.4 EM Algorithm Convergence Experiments

The standard estimation procedure for current phrase-based models relies on the generation of word alignment matrices [22]. In our proposal, such alignment matrices are generated by means of HMM-based word alignment models that are incrementally updated from user feedback. The necessity of incrementally update the models forces us to replace the batch EM algorithm by the incremental EM algorithm. Given the great importance of generating word alignments in the estimation of phrase-based models, we carried out experiments to compare the convergence rates of batch and incremental EM algorithms for HMM-based word alignment models.

Figure 1 shows the normalised log-likelihood that is obtained when executing 5 training epochs\(^3\) of the batch and incremental versions of the EM algorithm (common training schemes in state-of-the-art SMT systems frequently execute 5 EM training epochs to train the different word-alignment models). Results were obtained for the English–Spanish language pair of the Europarl corpus.

![Figure 1: EM convergence experiment comparing the normalised log-likelihood obtained when executing 5 training epochs of the batch and incremental versions of the EM algorithm. The experiments were executed on the Europarl English–Spanish corpus.](image)

According to the results presented in Figure 1, the incremental EM algorithm is able to obtain a greater normalised log-likelihood than that obtained by the batch EM algorithm for the two corpora under consideration. In addition to this, such a greater log-likelihood can be obtained with less EM training epochs. These observed results are due to the fact that the

\(^3\)An epoch is a single presentation of all samples in the training set.
incremental EM algorithm executes complete E and M steps for each training sample, resulting in a much greater rate of model updates per each training epoch [31].

In an online learning framework, whenever a new validated translation is available, it is used to extend the models and discarded afterwards. This means that only one training epoch of the incremental EM algorithm is performed when training HMM-based alignment models. This contrasts with the conventional batch training scheme, in which a few training epochs (typically 5) are executed. Hence, to fairly compare batch learning with our proposed online learning strategy, we should observe the relationship between the normalised log-likelihood of the incremental EM algorithm at the first training epoch and that of the fifth training epoch of the batch algorithm. According to the normalised log-likelihood values shown in Figure 1, we can appreciate a very small degradation in the log-likelihood (of less than 1% for the Europarl corpus).

1.2.5 Impact of Update Frequency

One important aspect to be clarified when designing PE or IMT systems is the influence of the system update frequency in the obtained performance. It is expected that updating the system in a sentence-wise manner will produce the best results. However, this updating strategy poses efficiency problems due to the necessity of executing model updates in real time. This problem can be alleviated by defining an alternative update strategy in which the training process is delayed until a certain number of samples has been gathered. Delaying model updates may cause a degradation in system performance, but it also constitutes one way to reduce the strong time requirements of a sentence-wise updating strategy. Specifically, if the time between updates is sufficiently high, the use of batch learning techniques could be appropriate (e.g. the training process can be executed overnight), removing the necessity of implementing online learning techniques.

To test the impact of update frequency in the system performance, we carried out IMT experiments using the English–Spanish Europarl corpus. In the experiments, the first 5,000 sentences of the different training sets were translated, using WER and KSMR to measure the user effort in the PE and IMT scenarios, respectively. The system was initialised with empty models and after that, such models were extended from the user validated translations using three different update frequencies, specifically, every 10, 100 and 1,000 sentences. Model updates were performed by means of conventional batch learning techniques, that is, the whole set of training samples seen so far is batch retrained whenever the model is updated. Additionally, we adopted default values for the weights of the log-linear model. To save computation time, the translations were generated by means of a monotonic decoder. The results of the experiments are shown in Figure 2.

As it can be observed in Figure 2, the user effort in terms of WER and KSMR was lower when the update frequency was increased. More specifically, batch retraining every 10 samples (Batch10) consistently outperformed the rest of the systems in all cases and retraining every 100 samples (Batch100) was also consistently better than retraining every 1,000 sentences (Batch1000). Sharper curves were obtained when translating the Xerox corpora, probably reflecting that in this corpus, there are groups of sentences with highly different translation difficulties from the system point of view.

1.2.6 Batch versus Online Learning

The great impact of frequent updates in the system performance demonstrated in the previous section poses the necessity of replacing conventional batch learning techniques by online learning techniques. EM convergence experiments provided in Section 1.2.4 showed that the
Figure 2: Impact of update frequency when translating the first 5,000 sentences of the English–Spanish language pair of the Europarl corpus. Plot shows the evolution of user effort measured in terms of KSMR as a function of the number of interactively translated sentences. A conventional SMT system was batch retrained every 10, 100 and 1,000 sentences. The system was initialised with empty models and default values for the weights of the log-linear model. Monotonic translations were generated.

Figure 3: Comparison between batch and online learning when interactively translating the first 5,000 sentences of the English–Spanish language pair of the Europarl corpora. The batch learning system was retrained every 10 sentences. The system was initialised with empty models and default values for the weights of the log-linear model.

As it can be seen in Figure 3, the performance of online learning is slightly better than that of batch learning. We think that the reason for this slight improvement is the higher update frequency of online learning, since the SMT models are extended for each individual training pair. Additionally, it should be noted that the shape of the curves obtained with online learning are very similar to that of batch learning.

1.2.7 Conclusions

During this second year of the project, part of the work presented within Task 4.1 has been focused on extending the experiments to measure the performance of online learning. The
presented results show the strong impact of update frequency in the performance of the ITP system. Ideally, the best system should be capable of updating its models in a sentence-wise manner. This update frequency cannot be achieved in real time by means of batch learning. On the other hand, we have compared the performance of online learning with that of batch learning, showing that online learning is able to achieve comparable or even slightly better results.

1.3 Distributed Implementation of Phrase-Based Model Estimation

The initially implemented online learning techniques were not able to deal with the very large corpora that are becoming more and more usual in the field of SMT. This includes the corpora present in the casmacat project. The first releases of the Thot toolkit [34], which is being used in the project, already included tools to estimate phrase-based models using the Map-Reduce technique [12], allowing the distributed execution of the training process in computer clusters. However, the word alignments that are required to extract the phrase pairs that compose the model were obtained by means of the well-know GIZA toolkit [32]. Such a toolkit does not currently include online estimation using the incremental version of the EM algorithm [31].

The Thot toolkit is being extended in different ways within the casmacat project, including a totally new tool to estimate HMM-based word alignment models by means of the incremental EM algorithm. Such a tool also applies the Map-Reduce technique to estimate models from corpora of an arbitrary size. The estimation process can be executed on clusters or multiprocessor systems, reducing the time costs. This new software has been successfully applied in different ITP experiments executed during the second year of the project, including those presented in Section 1.2. In addition to this, we expect to release a new version of the Thot toolkit including the above mentioned functionality before the end of the casmacat project.

1.4 Online Learning of the Log-Linear Weights

1.4.1 Introduction

The work performed during the last year concerning the online learning of the log-linear weights was further explored. The main problem observed during the first year of the period allotted for this task was that the improvements achieved by different techniques as applied in a conventional SMT setup did not carry on to the IMT framework, and sometimes even introduced losses in performance (see D4.1 for more details). In order to cope for this problem, the different algorithms for online learning of the weights were redefined, and some encouraging results were observed by obtaining weight samples from a Gaussian distribution and hence yielding different wordgraphs.

1.4.2 Development

During this year, we explored the possibility of sampling such weights by means of the Simplex algorithm. For this purpose, the optimum weight combination for each sentence was computed by means of such algorithm after the sentence had already been evaluated. Since the Simplex algorithm is an iterative algorithm, the different weight combinations obtained at each iteration constituted the weight sampling needed for building the different wordgraphs.

Experiments were conducted using the official casmacat corpora, and are reported in Table 2.

In this table, baseline displays the KSMR achieved without any online learning procedure enabled. original refers to the online learning of the log-linear weights as originally defined for
Table 2: Results in KSMR of the different online learning strategies studied

<table>
<thead>
<tr>
<th>Method</th>
<th>Weights</th>
<th>KSMR</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>—</td>
<td>40.6</td>
</tr>
<tr>
<td>original</td>
<td>—</td>
<td>42.8</td>
</tr>
<tr>
<td>gaussian</td>
<td>201</td>
<td>40.9</td>
</tr>
<tr>
<td>simplex</td>
<td>70</td>
<td>40.4</td>
</tr>
</tbody>
</table>

SMT (see D4.1). **gaussian** refers to the sampling strategy based on sampling from a Gaussian distribution (see D4.1). Finally, **simplex** refers to the strategy described above. As shown, the best-performing strategy is **simplex**. However, it must be noted that the improvements achieved by **simplex** are very similar to those obtained during the first year of this task with the **gaussian** strategy. However, there are three differences between those results and the ones reported here for **gaussian**:

1. The results presented last year did not involve the official CASMACAT corpora.
2. The results reported last year involved 501 weights samplings, and the ones reported this year involve only 201. This was so because the official CASMACAT corpora are much bigger than the ones used to report results last year, which emans that experiments on the official CASMACAT corpora revealed to be computationally much more expensive.
3. Results were reported last year in terms of word-stroke ratio, whereas the results presented here are in terms of KSMR. This was done so as to employ the same evaluation metric than in the rest of this workpackage.

Additionally, and concerning point 2) above, one thing which is also important to note is that the Simplex procedure drew an average of 70 weight samplings per sentence. This means that the Simplex procedure achieved better results by using less computational resources.

1.4.3 Conclusions

The results obtained in the task of online learning of the log-linear weights were very positive when applied to a conventional SMT setup. Even though slight and encouraging gains were also obtained in an IMT setting, such mixed results obtained point towards the fact that more research would still be needed.

2 Active Learning for Interactive Translation Prediction

2.1 Introduction

Current state-of-the-art machine translation (MT) systems are still far from generating error-free translations [1, 29]. Indeed, they usually require human experts to post-edit their automatic translations. This serial process prevents MT systems from taking advantage of the knowledge of the human experts, and the users cannot take advantage of the adaptive ability of MT systems.

An alternative way to utilise the existing MT technologies is to use them in collaboration with human translators within a computer-assisted translation (CAT) framework [19], and, particularly interesting among them, it is the ITP approach. However, despite being an efficient CAT protocol, conventional ITP technology has two potential drawbacks. First, the user is
required to supervise all the translations. Each translation supervision involves the user reading and understanding the proposed target language sentence, and deciding if it is an adequate translation of the source sentence. Even in the case of error-free translations, this process involves a non-negligible cognitive load. Second, conventional IMT systems consider static SMT models. This implies that after being corrected the system may repeat its errors, and the user will be justifiably disappointed.

2.2 Development

In [16], see Attachment A, we propose a cost-sensitive active learning (AL) framework for ITP whose goal is to make the translation process as efficient as possible. That is, we want to maximise the translation quality obtained per unit of user supervision effort. Note that this goal differs from the goal of traditional AL scenarios. While they minimise the number of manually-translated sentences to obtain a robust MT system, we aim at minimizing the number of corrective actions required to generate translations of a certain quality.

The proposed cost-sensitive AL framework boosts the productivity of ITP technology by addressing its two potential drawbacks. On the one hand, we do not require the user to exhaustively supervise all translations. Instead, we propose a selective interaction protocol where the user only supervises a subset of informative translations. This selective supervision is similar to the active interaction protocol studied in Task 2.3 in WP2 but applied at the sentence-level. On the other hand, we replace the batch SMT model typically used by ITP systems by an incremental SMT model [35] that utilises user feedback to continually update its parameters after deployment.

The potential user effort reductions of our proposal are twofold. On the one hand, user effort is focused on those translations whose supervision is considered most “informative”. Thus, we maximise the utility of each user interaction. On the other hand, the SMT model is continually updated with user feedback. Thus, the SMT model is able to learn new translations and to adapt its outputs to match the users preferences which prevents the user from making repeatedly the same corrections.

2.3 Conclusions

We evaluated the proposed cost-sensitive AL framework in a simulated translation of real data. Results showed that the use of user-supervised translations reduced to one half the effort required to translate the data. Additionally, the use of an adequate ranking function to implement the selective sampling further improved translation productivity.

3 Domain and User Adaptation

3.1 Introduction

Adaptation has become a very popular issue in natural language processing [27, 18, 23], and more specifically in statistical machine translation (SMT) [24]. Typically, the adaptation problem arises when two very different sets of training data are available, yielding two different sets of model parameters. The first set of data, the training data \( \mathcal{T} \) (e.g., obtained from the European Parliament or the United Nations) is often very large and rather generic in domain. The second set of data, the adaptation data \( \mathcal{A} \), belongs to the specific task of interest, such as printer manuals or medical diagnoses, and is usually overwhelmingly smaller than \( \mathcal{T} \). Then, the challenge is to modify the SMT system appropriately by taking into consideration both \( \mathcal{T} \) and
on the one hand, $T$ is ought to provide robustness in the estimation of the model parameters $\theta$, and on the other hand $A$ should introduce a certain bias towards the specific task.

This definition of adaptation is specially appropriate for the Bayesian learning paradigm, where the model parameters $\theta$ are treated as (hidden) random variables governed by some kind of a priori distribution $p(\theta)$. This distribution represents our prior knowledge about what values for $\theta$ should be good estimates. Estimating $p(\theta)$ by using a sufficiently large collection of data $T$ allows us to obtain a canonical model with parameters $\theta_T$, and it can be assumed that such estimation is a robust estimation. Then, as further evidence arrives in form of adaptation data $A$, we would like that such estimations are revised so that they reflect the newly arrived data. Considering $A$ within the Bayesian predictive distribution leads precisely to a scenario in which the decision regarding the output sentence includes a bias towards $A$, but is still guided by $p(\theta_T)$ (i.e., the prior distribution given $T$). Hence, under the Bayesian predictive adaptation (BPA) framework, the final translation is not computed by considering only the topic-specific data (i.e., $A$), which could lead to over-trained estimations of $\theta$: if the amount of data available is small, the parameter prior $p(\theta)$ will compensate this issue, providing robustness \cite{13}. However, the effect of this prior knowledge fades when incorporating further evidence, until a point in which the contribution of the parameter prior towards the complete model distribution is negligible. In addition, the Bayesian learning paradigm does not attempt to obtain a single best point estimate of $\theta$, but rather relies on considering all possible parameter values, allowing uncertainty regarding what the best estimations of such parameters might be.

In the work conducted so far, we focus on the Bayesian adaptation of the weights of the log-linear combination of features present in every state-of-the-art SMT system. Even though these weights are not very numerous (generally in the range of 10 or 20), providing the system with appropriate estimates for these weights is critical \cite{10}. We propose a Bayesian approach in which the adaptation data is taken into account when computing the posterior over the model parameters (i.e., the log-linear weights). In addition, including a Gaussian prior within this posterior ensures a certain stability of the estimates, leading to more reliable predictions.

3.2 Development

We first introduce the typical formulation of SMT \cite{24}. In state-of-the-art SMT systems, it is quite common to have a log-linear combination of features $h$, weighted by scaling factors $\lambda$. Then, the probability of the output sentence $y$ given the input sentence $x$ is computed as

$$p(y \mid x) = \frac{\exp \sum_{m} \lambda_m h_m(x, y)}{\sum_{y'} \exp \sum_{m} \lambda_m h_m(x, y')}$$  \hspace{1cm} (6)

and the decision rule is given by the expression

$$\hat{y} = \arg \max_y \sum_{m} \lambda_m h_m(x, y).$$  \hspace{1cm} (7)

Typically, the weights $\lambda$ of the log-linear combination are estimated on a development set by means of error-driven algorithms such as MERT \cite{33} or MIRA \cite{9}, which have proven to provide good estimates if the amount of data available is sufficient and the characteristics of the data to be translated match approximately those of the development set. However, if either of these two premises are not fulfilled, over-fitting to the specific characteristics of the development set occurs and such algorithms fail to provide appropriate estimates \cite{13,10}.

In this article, we propose to reformulate the decision rule whenever the following conditions are met:

4For readability purposes, we directly instantiate the model parameters $\theta$ to the parameters we intend to adapt, i.e., $\lambda$. 

14
• A development set for a given “old” domain is available, or a canonical estimation of $\lambda$ is readily available.
• The text to be translated belongs to a different “new” domain.
• A small set of development data is available for the new domain, but such set is insufficient for a proper estimation of $\lambda$.

The meaning of “large” in this context depends mainly on the target domain, but also on the estimation method to be used. This will be analysed more in depth in the experiments section.

Under the circumstances described above, we consider adapting $\lambda$, instead of performing a full re-estimation. For this purpose, we propose the use of the Bayesian paradigm \cite{5}, in which parameters are viewed as random variables with some kind of underlying distribution. Considering $T$ as the training data, and $A$ as an additional adaptation set, Equation \ref{eq:6} is rewritten by means of the predictive distribution as

$$p(y \mid x; T, A) = \int p(y, \lambda \mid x; T, A) d\lambda$$

$$\approx \int p(\lambda \mid T, A) p(y \mid x, \lambda) d\lambda.$$  \hfill (8)

From Equation \ref{eq:6} to Equation \ref{eq:9} it has been assumed that the probability of the output sentence $y$ does not depend on $A$ and $T$, whenever the model parameters $\lambda$ are known. It has also been assumed that $\lambda$ is independent from the actual input sentence $x$. Such simplifications lead to a decomposition of the integral in two parts: the first one, $p(\lambda \mid T, A)$, will assess how good the model parameters are, and the second one, $p(y \mid x, \lambda)$, will account for the quality of the translation $y$ given $\lambda$. The integral will force the model to take into account all possible parameter values, although the parameter prior will bias the final distribution towards our prior knowledge.

Operating with the probability of $\lambda$, we obtain:

$$p(\lambda \mid T, A) = \frac{p(A \mid \lambda; T) p(\lambda \mid T)}{\int p(A \mid \lambda'; T) p(\lambda' \mid T) \, d\lambda'}.$$  \hfill (10)

In order to simplify Equation \ref{eq:10} and focusing on the probability of the adaptation data $A$, of a given size $|A|$, we obtain:

$$p(A \mid \lambda; T) \approx p(A \mid \lambda) = \prod_{a=1}^{|A|} p(x_a \mid \lambda) \, p(y_a \mid x_a, \lambda),$$  \hfill (11)

where the probability of $A$ has been assumed to be independent of $T$, given that $\lambda$ is known, and has been modelled as the probability of each bilingual sample $(x_a, y_a) \in A$ being generated independently by a given translation model.

For modelling the prior over the model parameters, i.e., $p(\lambda \mid T)$, we will assume that $\lambda$ follows a normal distribution centred on $\lambda_T$, i.e., the parameter values estimated on the training data ($T$), and with a diagonal covariance matrix $I \cdot \sigma_T$ with variance $\sigma_T$ bounded for all parameters, yielding

$$p(y \mid x; T, A) \approx Z \int p(A \mid \lambda; T) p(\lambda \mid T) \, p(y \mid x, \lambda) \, d\lambda$$

$$\approx Z \prod_{a=1}^{|A|} p(y_a \mid x_a, \lambda) N(\lambda; \lambda_T, I \cdot \sigma_T) p(y \mid x, \lambda) \, d\lambda.$$  \hfill (12)
Here, $Z$ is the normalisation constant required for ensuring that $p(y \mid x; \mathcal{T}, A)$ defines a probability distribution. The term $p(x_a \mid \lambda)$ present in Equation 11 can be simplified if $p(A \mid \lambda; T)$ is plugged into Equation 10 and if $x_a$ can be assumed independent of $\lambda$.

Plugging in the log-linear model described in Equation 6, we obtain

$$p(y \mid x; \mathcal{T}, A) \propto Z \prod_{a=1}^{\vert A \vert} \exp \sum_m \lambda_m h_m(x_a, y_a) \left( \sum_{y'} \exp \sum_m \lambda_m h_m(x_a, y') \right)$$
$$\times \exp \left\{ -\frac{\vert\lambda - \lambda_T\vert^2}{2 \sigma_T} \right\} \sum_{y'} \exp \sum_m \lambda_m h_m(x, y') \ d\lambda,$$

and, finally, the decision rule will be given by the maximisation of the previous equation, i.e.,

$$\hat{y} = \arg \max_y p(y \mid x; \mathcal{T}, A).$$

Note that, in this case, the denominators in Equation 13 cannot be easily neglected, as was the case in Equation 7 since they are affected by the integral.

In the next section, Markov-chain Monte-Carlo is presented for approximating the integral in Equation 13. However, before carrying on with its presentation, there are several approximations that need to be performed so that the predictive distribution can be computed. Firstly, $p(A \mid \lambda; T)$ and $p(y \mid x, \lambda)$ contain in their denominator sums over all possible sentences of the target language, which is not computable. For this reason, $\sum_{y'}$ is approximated as the sum over all the hypothesis within the $n$-best list generated during the regular decoding process. Coherently, instead of performing a full search of the best possible translation we will only consider eligible the ones present in such $n$-best list. In addition, typical state-of-the-art SMT systems do not guarantee complete coverage of all possible sentence pairs due to the great number of heuristic decisions involved, and out-of-vocabulary words may imply that the SMT model is unable to explain a certain bilingual sentence completely. Hence, instead of using the true reference present in $A$, we will use the best possible translation $y^*$ generated during the decoding process. $y^*$ is often referred to as oracle derivation in related work [9].

The formulation presented would also allow considering as model parameters the feature functions $h(\cdot, \cdot)$. However, in the present article we will only consider the adaptation of $\lambda$, since adapting $h$ is much more costly and is left as future work.

### 3.3 Markov chain Monte Carlo

Computing the integral over the complete parametric space, as described in Equation 13 is often unfeasible from the computational point of view. Moreover, it may also be the case that the function to be integrated is not even integrable. Hence, it is often approximated by a discrete sum over a sampling of such parameters. For simplicity $\mathcal{S}(\lambda_T)$ will denote a specific sampling of $\lambda$.

Markov chain Monte Carlo (MCMC) methods [5] obtain samples $\mathcal{S}(\lambda_T)$ of a variable (in this case $\lambda$) assumed to follow a certain distribution, i.e., $p(\lambda \mid \mathcal{T}, A)$. MCMC methods are specially suited for sampling distributions where the normalisation constant cannot be evaluated [5]. For doing this, a (first order) Markov chain is established, where each new sample $\lambda^*$ depends on the previous sample $\lambda'$. Specifically, in this article we will be using the Metropolis-Hastings (MH) algorithm [17], which consists in drawing a sample $\lambda^*$ from a given proposal distribution $q(\lambda \mid \lambda')$. Then, $\lambda^*$ is accepted with probability

$$A(\lambda^*, \lambda') = \min \left( 1, \frac{\tilde{p}(\lambda^*) q(\lambda' \mid \lambda^*)}{\tilde{p}(\lambda') q(\lambda^* \mid \lambda')} \right),$$

16
with \( p(\lambda) = \tilde{p}(\lambda) / Z_p \) being the distribution from which we intend to sample, and \( Z_p \) being the normalisation term for \( p(\lambda) \). In Equation 15, \( \tilde{p}(\lambda) \) can be safely used instead of \( p(\lambda) \), since \( Z_p \) would be simplified. If the proposal distribution is symmetric, terms \( q(\cdot | \cdot) \) can also be simplified.

The proposal distribution is often set as a normal distribution \( \mathcal{N}(\lambda; \lambda', I \cdot \sigma_o) \), with mean vector \( \lambda' \) and covariance matrix a diagonal matrix with main diagonal \( \sigma_o \). Establishing \( \sigma_o \) is critical, since too small values will lead to a high rejection rate and the sampling chain will most likely get stuck at a local maximum, while too big values will lead to a chaotic chain which will not sample the density function appropriately.

Another aspect that needs to be taken into account when building a MCMC chain is the burn-in phase \[5\], which is the number of samples that need to be drawn in order to assume independence from the initial state of the Markov chain.

Once an appropriate sample \( S(\lambda_T) \) has been obtained from \( p(\lambda | T, A) \) (or, dropping the normalisation constant in Equation 10 from \( p(A | \lambda; T)p(\lambda | T) \)), Equation 12 is approximated, again according to the Strong Law of Large Numbers \[5\], as

\[
p(y | x; T, A) \approx Z' \sum_{\lambda \in S(\lambda_T)} p(y | x, \lambda),
\]

where \( \delta \) is not required either. Although the right hand of Equation 16 seems independent from \( T \) and \( A \), this is only a notation issue, since such dependency is hidden within \( S(\lambda_T) \), and \( S(\lambda_T) \) must be recomputed for every adaptation set \( A \).

### 3.4 Experiments

Experiments were performed by means of the open-source MT toolkit Moses \[26\] (version 0.91) in its default non-monotonic configuration, which includes 5 translation models (direct- and inverse- translation and lexicalised models and the phrase-penalty), 7 re-ordering models (an exponential model and the six models included within the msd-reordering-fe model \[25\]), the word- penalty and a word-based language model, i.e., \(|\lambda| = 14\). The language model used was a 5-gram with modified Kneser-Ney smoothing \[20\], built with the SRILM toolkit \[39\].

Translation quality was assessed mainly by means of single-reference TER \[35\]. TER (Translation Error Rate) is an error metric that computes the minimum number edits required to modify the system hypotheses so that they match the reference. Possible edits include insertion, deletion and substitution of single words, as well as shifts of word sequences. Some results will also be presented in terms of BLEU \[36\], with the purpose of assessing whether the improvements in TER also correspond to improvements in BLEU. BLEU (BiLingual Evaluation Understudy) is a precision metric that measures n-gram (\( n \leq 4 \)) coverage of the system hypotheses with respect to the reference, with a penalty for sentences that are too short.

For computing the best possible hypothesis \( y^* \), TER will be used, since BLEU is not always well defined at the sentence level, given that it implements a geometrical average which is zero whenever there is no common 4-gram between hypothesis and reference, e.g., a 3-word sentence. Selecting \( y^* \) with smoothed versions of BLEU is planned as future work.

In the figures of this section, the points in the plots presented display the average of ten experiments, in which the adaptation data \( A \) was re-drawn each time with replacement. Unless stated otherwise, the x-axis will always be in logarithmic scale and display \(|S(\lambda_T)| \). The scale of the y-axis will be linear whenever the plot displays translation quality (TER unless stated otherwise), and logarithmic in the case of the confidence interval sizes (in TER points

\[5\]Available from http://www.statmt.org/moses/
unless stated otherwise). These confidence intervals present the 95% confidence level and were computed as $2\sigma$, where $\sigma$ is the empirical standard deviation observed in the 10 repetitions. Note that the full confidence interval would be $4\sigma$, i.e., $\pm 2\sigma$. Confidence intervals are displayed in different plots, instead of using error bars, because otherwise the translation quality plots would present vertical lines across the complete plot, rendering it unreadable. For analysing the effect of the different meta-parameters in BPA, we performed experiments on all the available domains (explained in next section). Although the resulting plots were very similar for all corpora analysed, we only present here the clearest plot, for readability purposes.

### 3.5 Corpora

The experiments conducted in this work were carried out on three different corpora, belonging to different domains, all of them stemming from the domain adaptation Summer workshop carried out at the John Hopkins University in 2012 [7]. In this workshop, the task was to adapt French $\rightarrow$ English translation models. The out-of-domain corpus provided originated in the parliamentary domain (Canadian Hansards), and the in-domain corpora included the medical domain (henceforth referred to as EMEA), and the general news domain (henceforth referred to as NEWS). Statistics of the out-of-domain corpus are provided in Table 3 and statistics of the in-domain corpora are provided in Table 4. Note that this task was intended for an adaptation problem involving many more parameters, i.e., feature function adaptation, which requires much more data than the problem of scaling factor adaptation. Since the amount of scaling factors to be adapted is typically 14, we will only make use of much less data, since it is common knowledge that scaling factors are already well estimated with a development of about 2000 sentences. However, the amount of data available allows us to perform several random extractions of the adaptation set $A$, allowing for a more robust analysis of the results obtained and being able to compute confidence intervals of the translation quality achieved by the different methods studied.

<table>
<thead>
<tr>
<th></th>
<th>Training ($T$)</th>
<th>Development ($D$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>French</td>
<td>English</td>
</tr>
<tr>
<td>Hansards</td>
<td>8.1M</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td>163M</td>
<td>144M</td>
</tr>
<tr>
<td></td>
<td>191.4k</td>
<td>186.8k</td>
</tr>
</tbody>
</table>

Table 3: Main figures of the out-of-domain corpus. $OoV$ stands for Out-of-Vocabulary words (types) with respect to the training data ($T$).

<table>
<thead>
<tr>
<th></th>
<th>Training ($A$)</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>French</td>
<td>English</td>
</tr>
<tr>
<td>EMEA</td>
<td>472k</td>
<td>2045</td>
</tr>
<tr>
<td></td>
<td>6.5M</td>
<td>5.9M</td>
</tr>
<tr>
<td></td>
<td>35k</td>
<td>30k</td>
</tr>
<tr>
<td>NEWS</td>
<td>136k</td>
<td>2489</td>
</tr>
<tr>
<td></td>
<td>3.9M</td>
<td>3.3M</td>
</tr>
<tr>
<td></td>
<td>28.9</td>
<td>24.6</td>
</tr>
</tbody>
</table>

Table 4: Main figures of the in-domain corpora. $OoV$ stands for Out-of-Vocabulary words (types) with respect to the out-of-domain corpus ($T$).

The standard features $h$ were estimated on the training partition of the Hansards corpus, whereas a canonical $\lambda_T$ was estimated on the development subset $D$ (i.e., a held-out subset of
the training set $\mathcal{T}$ used to estimate $h$) by means of the default MERT implementation within Moses. Translation quality will be estimated on the different in-domain test subsets.

### 3.6 Comparison between BPA and parameter re-estimation

To synthesise the different strategies, the different SMT systems compared in this section when adapting $\lambda$ are:

- **Baseline system**: Phrase-pairs extracted from the Hansards training corpus, i.e., $h$ estimated on $\mathcal{T}$. Scaling factors $\lambda_{\mathcal{T}}$ estimated on $\mathcal{D}$ by means of MERT.

- **MCMC**: Initial setup identical to the baseline system. Adaptation samples $\mathcal{A}$ were randomly extracted from the training partitions of the in-domain corpora (i.e., EMEA or NEWS). $\lambda_{\mathcal{T}}$ within $p(\lambda \mid \mathcal{T})$ estimated on $\mathcal{D}$ with MERT. The sampling strategy used within BPA was MCMC.

- **MERT**: Initial setup identical to the baseline system. Then, the adaptation samples $\mathcal{A}$ described above were used for estimating a new set of scaling factors using MERT.

- **MIRA**: As MERT, but scaling factors are estimated by means of MIRA. Note that MIRA is an incremental estimation strategy, and $\lambda_{\mathcal{T}}$ were provided for the MIRA starting point. This means that MIRA could also be considered a sort of adaptation strategy (i.e., not full re-estimation).

- **PRO**: As MERT, but scaling factors are estimated by means of pairwise optimisation.

In addition, we also conducted experiments by concatenating $\mathcal{A}$ and $\mathcal{D}$, and using the result for estimating $\lambda$ by means of MERT. However, such strategy performed consistently worse in terms of TER than the other re-estimation strategies analysed here. For this reason, this setup was removed from the final comparison in order to avoid clogging the plots with too many curves.

It must be emphasised, however, that the re-estimation strategies are not really a fair comparison, since they are all by far much more costly than BPA. In addition, they involve several translation steps, each of which re-computes the $n$-best list, and have better chances to obtain better hypotheses, whereas the BPA strategies implemented rely on a pre-computed $n$-best list of fixed size (in this Section, $n$-best size was set to 500).

Results of such comparison can be seen in Figure 4. There are several things that should be noted:

- For small amounts of adaptation data, BPA is the strategy that performs the best in all cases except for the NRC corpus.

- MERT and PRO display a very unstable behaviour for small $|S(\lambda_{\mathcal{T}})|$ sizes, and MIRA seems to exhibit better performance. This is not surprising, since MIRA can also be seen as an adaptation strategy because it uses $\lambda_{\mathcal{T}}$ as starting point.

- The SUBS corpus appears to be a specially difficult corpus. In the first place, TER scores are specially high. In addition, neither the MERT and MIRA strategies are not able to improve over the unadapted baseline.
Figure 4: Performance comparison across the different corpora analysed and with the different \( \lambda \) estimation strategies. The two plots on the left display TER, while the two plots on the right display the size of the confidence intervals.

- For small amounts of adaptation data, the confidence intervals of PRO, MIRA and MERT may be as big as 10-20 TER points. When taking into account a baseline of about 65 TER points, 10 points may well imply the difference between the output being useful or being completely useless. In contrast, BPA seldom yields confidence intervals of more than 2 TER points. We consider this point important, since the stability achieved by BPA reveals precisely that BPA is an appropriate adaptation strategy for this problem: the size of the confidence intervals can be seen as a measure of how prone is a given algorithm to over-training, and BPA proves to be able to provide quite stable estimations even for very small amounts of data.

From these observations, it can be concluded that BPA is an effective adaptation strategy. An adaptation strategy only is useful when the amount of adaptation data is small, and BPA proves to perform well under such circumstances. If the amount of adaptation data is larger (> 100), BPA still yields an acceptable behaviour, although the pure re-estimation strategies do yield better estimates of \( \lambda \).

Since BLEU is a more standard evaluation metric than TER among the SMT community, we also report some BLEU results in Figure 5. The overall analysis varies little with respect to the one done with TER. However, two things are noteworthy: firstly, BPA tends to perform slightly worse in terms of BLEU. This is actually expected, since the best possible hypothesis \( y^* \) is selected according to TER, and BLEU includes a brevity penalty which TER does not take into account. Second, the size of the confidence intervals obtained is nearly the same as in the case of TER, and BPA tends to achieve smaller confidence intervals, specially in the case of small amounts of adaptation data. It is also interesting to see that the re-estimation
Figure 5: Performance comparison across the different corpora analysed and with the different λ estimation strategies. The two plots on the top display BLEU, while the two plots at the bottom display the size of the confidence intervals.

approaches need more than 100 adaptation samples in order to achieve the performance that the heuristic version of BPA achieves with only 10 samples. As in the case of the experiments involving TER, the results for NEWS and NRC were similar to those obtained with the EMEA corpus, and SUBS seems to be specially difficult.

### 3.7 Conclusions

During this first year of work, Bayesian predictive adaptation has been analysed for its application to log-linear weight adaptation in statistical machine translation. On the one hand, the theoretical framework for adapting the scaling factors present in most state-of-the-art SMT systems has been developed. On the other hand, experimental results analysing the effectiveness of such adaptation procedures have been reported.

Results show that BPA is able to provide consistent improvements in translation quality over the baseline systems, as measured by TER, with as few as 10 adaptation samples, and up to an amount of adaptation data that allows a complete re-estimation of the model parameters. In addition, BPA proves to be more stable than most re-estimation strategies, which rely heavily on the amount of adaptation data. It should be emphasised that an adaptation technique, by nature, is only useful whenever the amount of adaptation data is low, and BPA proves to behave well in such context. Whenever the amount of adaptation data is high, the best thing that one can do is to re-estimate the model parameters from scratch, although such re-estimation is often very costly. From a computational point of view, the Bayesian adaptation technique presented does not imply a significant computational overhead, and most terms can be precomputed in the case of heuristic sampling. Hence, we consider that it could be easily implemented within
the decoder itself without a significant increase in computational complexity. Nevertheless, it must be taken into account that the search space explored by a given \( n \)-best list is much more restrained that the one that the decoder will take into account. This means that, if BPA is to be implemented within the decoder (instead of by re-scoring \( n \)-best lists), the amount of \( n \)-best considered by BPA in the term \( p(A \mid \lambda; T) \) must be sufficiently large. We plan to explore this in future work.

3.8 Future work

Concerning the BPA approach, we are researching an incremental version of the techniques described here, with the purpose of applying BPA within an online adaptation scenario. We also plan to analyse the possibility of adapting the log-linear features of the translation model, as well as applying BPA in a more feature-rich SMT model.

In other terms, we also plan to study other adaptation approaches which are present in the literature and with which we already have experience, such as:

1. Sentence selection [15]. The main purpose of this strategy is to select, from the generic domain corpora, which of the sentences are best suited for training an SMT system for the specific domain tackled. Similar strategies have already reported substantial gains, both in lab experiments and field trials, and we believe that they should also provide improvements in the CASMACAT framework.

2. Language model interpolation [37]. The BPA approach described above has been designed for the adaptation of the parameters of the translation model. Even though similar formulation could be devised for the adaptation of the language model, we already have experience by using language model interpolation. We also plan to research the applicability of such strategy to the CASMACAT framework.

3. Translation model interpolation [23]. Building different translation tables and interpolating them according to the specific domain to be translated is one of the most standard adaptation strategies present in the literature. Hence, we also plan to explore this during the third year of the CASMACAT project.
References


Attachment A

Task 4.2

Jesús González-Rubio and Francisco Casacuberta.

Cost-sensitive active learning for computer-assisted translation.

*Pattern Recognition Letters, 2013.*
Cost-sensitive active learning for computer-assisted translation

Jesús González-Rubio a,⇑, Francisco Casacuberta b

a Institut Tecnològic d’Informàtica, Universitat Politècnica de València, Camino de Vera s/n, 46022 València, Spain
b D. Sistemes Informàtics i Computació, Universitat Politècnica de València, Camino de Vera s/n, 46022 València, Spain

Article info
Article history:
A R T I C L E I N P R E S S

Machine translation technology is not perfect. To be successfully embedded in real-world applications, it must compensate for its imperfections by interacting intelligently with the user within a computer-assisted translation framework. The interactive–predictive paradigm, where both a statistical translation model and a human expert collaborate to generate the translation, has been shown to be an effective computer-assisted translation approach. However, the exhaustive supervision of all translations and the use of non-incremental translation models penalizes the productivity of conventional interactive–predictive systems.

We propose a cost-sensitive active learning framework for computer-assisted translation whose goal is to make the translation process as painless as possible. In contrast to conventional active learning scenarios, the proposed active learning framework is designed to minimize not only how many translations the user must supervise but also how difficult each translation is to supervise. To do that, we address the two potential drawbacks of the interactive–predictive translation paradigm. On the one hand, user effort is focused to those translations whose user supervision is considered more “informative”, thus, maximizing the utility of each user interaction. On the other hand, we use a dynamic machine translation model that is continually updated with user feedback after deployment. We empirically validated each of the technical components in simulation and quantify the user effort saved. We conclude that both selective translation supervision and translation model updating lead to important user-effort reductions, and consequently to improved translation productivity.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Machine translation (MT) is a fundamental technology that is emerging as a core component of natural language processing systems. A good example of multilingualism with high translation needs can be found in the European Union (EU) political institutions. According to EC (2009), the EU employs 1,750 full-time translators. Additionally, to cope with demand fluctuations, the EU uses external translation providers which generate approximately one fourth of its translation output. As a result, in 2008 the EU translation services translated more than 1,800,000 pages and spent about one billion Euros on translation and interpreting.

Besides being an expensive and time-consuming task, the problem with translation by human experts is that the demand for high-quality translation has been steadily increasing, to the point where there are just not enough qualified translators available today to satisfy it. This poses a high pressure on translation agencies that must decide how to invest their limited resources (budget, manpower, time, etc.) to generate translations of the maximum quality in the most efficient way.

To address this challenge, many translation agencies have focused their interest on MT technology. However, current state-of-the-art MT systems are still far from generating error-free translations (NIST, 2006; Lopez, 2008). Indeed, they usually require human experts to post-edit their automatic translations. This serial process prevents MT systems from taking advantage of the knowledge of the human experts, and the users cannot take advantage of the adaptive ability of MT systems.

An alternative way to utilize the existing MT technologies is to use them in collaboration with human translators within a computer-assisted translation (CAT) framework (Isabelle and Church, 1998). An important contribution to CAT technology was carried out during the TransType project (Foster et al., 1998; Langlais et al., 2000; Foster, 2002; Langlais and Lapalme, 2002). They proposed the interactive–predictive machine translation (IMT) framework where data-driven MT technologies are embedded within the translation environment. Following these ideas, Barrachina et al. (2009) proposed an innovative embedding where a fully-fledged statistical MT (SMT) system is used...
to produce complete translations, or portions thereof, which can be accepted or amended by a human expert, see Fig. 1. Each corrected text segment is then used by the SMT system as additional information to achieve further, hopefully improved, translations.

Despite being an efficient CAT protocol, conventional IMT technology has two potential drawbacks. First, the user is required to supervise all the translations. Each translation supervision involves the user reading and understanding the proposed target language sentence, and deciding if it is an adequate translation of the source sentence. Even in the case of error-free translations, this process involves a non-negligible cognitive load. Second, conventional IMT systems consider static SMT models. This implies that after being corrected the system may repeat its errors, and the user will be justifiably disappointed.

We propose a cost-sensitive active learning (AL) (Angluin, 1988; Atlas et al., 1990; Cohn et al., 1994; Lewis and Gale, 1994) framework for CAT where the IMT user-machine interaction protocol (Fig. 2) is used to efficiently supervise automatic translations. Our goal is to make the translation process as efficient as possible. That is, we want to maximize the translation quality obtained per unit of user supervision effort. Note that this goal differs from the goal of traditional AL scenarios. While they minimize the number of manually-translated sentences to obtain a robust MT system, we aim at minimizing the number of corrective actions required to generate translations of a certain quality.

The proposed cost-sensitive AL framework boosts the productivity of IMT technology by addressing its two potential drawbacks. First, we do not require the user to exhaustively supervise all translations. Instead, we propose a selective interaction protocol where the user only supervises a subset of “informative” translations (González-Rubio et al., 2010). Additionally, we test several criteria to measure this “informativeness”. Second, we replace the batch SMT model by an incremental SMT model (Ortiz-Martínez et al., 2010) that utilizes user feedback to continually update its parameters after deployment. The potential user effort reductions of our proposal are twofold. On the one hand, user effort is focused on those translations whose supervision is considered most “informative”. Thus, we maximize the utility of each user interaction. On the other hand, the SMT model is continually updated with user feedback. Thus, the SMT model is able to learn new translations and to adapt its outputs to match the user’s preferences which prevents the user from making repeatedly the same corrections.

The remainder of this article is organized as follows. First, we briefly describe the SMT approach to translation, and its application in the IMT framework (Section 2). Next, we present the proposed cost-sensitive AL framework for CAT (Section 3). Then, we show the results of experiments to evaluate our proposal (Section 4). Finally, we summarize the contributions of this article in Section 5.

![Fig. 1. Diagram of an interactive–predictive MT system. To translate a source sentence \( x \), the user interacts with the system accepting or correcting the proposed translations \( y \). User feedback \( k \) is used by the system to improve its suggestions.](image)

![Fig. 2. IMT session to translate a Spanish sentence into English. At interaction-0, the system suggests a translation \( y_s \). At interaction-1, the user moves the mouse just before the first error and implicitly validates the first eight characters “To view” as a correct prefix \( y_p \). Then, the user introduces a correction by pressing the a key \( k \). Lastly, the system suggests completing the translation from the user correction with “list of resources” \( y_s \). At interaction 2, the user validates “To view a list” and introduces a correction \( k \) which is completed by the systems to form a new translation “To view a listing of resources”. Interaction 3 is similar. Finally, the user accepts the current translation which is equal to the desired translation.](image)
2. Interactive–predictive machine translation

The statistical machine translation (SMT) approach considers translation as a decision problem, where it is necessary to decide upon a translation \( y \) given a source language sentence \( x \). Statistical decision theory is used to select the correct translation among all the target language sentences. From the set of all possible target language sentences, we are interested in that with the highest probability according to the following equation (Brown et al., 1993):

\[
y = \arg \max_y \Pr(y|x)
\]  
(2.1)

where \( \Pr(y|x) \) is usually modeled by a maximum entropy MT model (Och and Ney, 2002), also known as log-linear model. The decision rule for log-linear models is given by the expression:

\[
y = \arg \max_y \Pr(y|x) \approx \arg \max_y \left\{ \sum_{m=1}^{M} \alpha_m h_m(y, x) \right\}
\]  
(2.2)

where each \( h_m(y, x) \) is a feature function that describes a particular aspect of the translation process (e.g. the log-probability \( \log(\Pr(y)) \) of the translation), and \( \alpha_m \) is its associated weight. Phrase-based (Koehn et al., 2003) and finite state (Casacuberta and Vidal, 2007) models are two successful implementations of the log-linear approach.

However, despite a huge research effort, SMT systems are still not perfect. To obtain high-quality translations, a human expert has to supervise the automatically generated translations. This supervision is usually carried out as a separate post-edition step. The IMT framework (Barrachina et al., 2009) constitutes an alternative to this serial procedure. In an IMT system, an SMT model and a human expert collaborate to generate error-free translations. These translations are generated in a series of interactions between the SMT model and the user. At each interaction, the SMT model generates a translation of the source sentence which can be partially or completely accepted and corrected by the user. Each partially corrected text segment, called prefix, is then used by the SMT model as additional information to generate better translations. These translations are then available to the user to use as corrections for translations that require the SMT models for their use at character level are trivial.

3. Cost-sensitive active learning for computer-assisted translation

Although IMT have been successfully deployed in many practical applications, it still demands the human user to supervise all translations. This exhaustive supervision guarantees that the generated translations are error-free. However, it demands a large amount of cognitive effort by the user which penalizes translation productivity. A translation agency with limited resources, in terms of person-hours, may be willing to sacrifice some translation quality in exchange for improved productivity. Certainly, this is an unrealistic scenario in some cases, for example it is inconceivable not to fully-supervise the translation of a legal document such as a contract, but there are many other translation tasks, e.g. manuals for electronic devices, or twitter and blog postings, that match this productivity-focused scenario.

The goal of this section is to present a cost-efficient CAT framework that allows the user to supervise and correct automatic translations as effortlessly as possible. From the existing IMT technology, we import its user-machine interaction process (Fig. 2) to efficiently supervise individual translations. However, we implement a different work-flow to address its drawbacks. On the one hand, user effort will be focused to supervise only those translations considered most “informative”. On the other hand, the translation model will be continually updated with the new sentence pairs \((x, y)\) supervised by the user.

We implement these ideas as a cost-sensitive AL scenario designed to minimize supervision effort, Section 3.1. We define a new translation work-flow, Section 3.2, that focuses user-effort to only supervise the subset of most “informative” translations. Section 3.3 describes the different ranking functions implemented to measure the “informativeness” of each translation, and finally, Section 3.4 presents the incremental SMT model that is continually updated from user feedback.

3.1. Active learning for computer-assisted translation

Training an SMT model requires translation examples of source language sentences and its corresponding target language translations. Example annotation is difficult for structured prediction tasks, since each example may have multiple, interacting labels, all of which must be correctly annotated for the example to be of use to the learner. This is particularly true for translation where additionally there may be multiple correct translations for a source sentence.

Different alternatives to conventional supervised learning have been proposed to address these problems. For example, semi-supervised learning methods use unlabeled data to help supervised learning tasks (Chapelle et al., 2006). These methods typically assume that the labeled data set is given and fixed. In practice, however, semi-supervised methods are allowed to pick a set of unlabeled examples to be annotated by an expert. In this case, rather than selecting the examples randomly, it may be attractive to let the learning algorithm to proactively tell us which of them to annotate. This approach is known as active learning (AL). The idea is to select which training examples to label and the order in which they are labeled to increase learning efficiency (Angluin, 1988; Atlas et al., 1990; Cohn et al., 1994; Lewis and Gale, 1994). An active learner is considered successful if it obtains better performance than a traditional learner given the same number of training examples. Therefore, AL exploits annotation by reducing the number of labeled examples required to train an accurate model.
In contrast to previous applications of AL to structured prediction tasks, e.g. sequence labeling (Settles and Craven, 2008), natural language parsing and information extraction (Thompson et al., 1999), or machine translation (Haffari et al., 2009), that minimize the number of labeled samples required to train an accurate model, our goal is to reduce the user supervision effort required to generate high-quality translations. Clearly, the amount of work required to supervise a translation will vary between sentences, e.g. based on the size and the complexity of the source sentence. Thus, it is desirable to design an AL supervision scenario that considers not only how many translations the user is required to supervise, but also how difficult each translation is to supervise.

### 3.2. Translation work-flow and supervision protocol

The proposed AL framework for CAT implies a modification of the conventional IMT work-flow depicted in Fig. 1. The user no longer supervises the translation of all sentences but only of those selected as “worthy of being supervised”. Since only the most informative sentences are supervised, we maximize the utility of each user interaction. Final translations however may not be error-free as for conventional IMT. In exchange, an important reduction in human effort is potentially achievable. Moreover, we can adapt the system’s behavior to the requirements of each particular translation task.

Conventional IMT technology is built over the implicit assumption that the inbound text to be translated behaves as a text stream (see Fig. 1). Source sentences are translated separately and no information is stored (or assumed) about the preceding (or following) sentences, e.g. how many sentences remain untranslated. Since the IMT framework uses static SMT models and requires the user to supervise all translations, this is not a strong assumption. However, we have to take it into account because information about previously supervised translations, and particularly, about following sentences may have great impact on the final user effort.

We handle the inbound text stream by partitioning the data into blocks of consecutive sentences. Within a block, all sentences are available, but once the algorithm moves to the next block, all sentences in previous blocks become inaccessible. We use the sentences within a block to estimate the current distribution of sentences in the stream, so that the estimation of the “informativeness” of supervising the translation of a sentence can be done as accurately as possible.

Algorithm 1 shows the pseudo-code that implements the proposed cost-sensitive AL scenario for CAT. The algorithm takes as input a stream of source sentences, a “base” SMT model, and an effort level $\rho$ denoting the percentage of sentences of each block to be supervised. First, the next block of sentences $B$ is read from the data stream (line 3). For each sentence in $B$, the current SMT model generates an initial translation, $\hat{y}$ (line 6). If the sentence has been sampled as worth of supervision, the user collaborates with the system to translate the sentence (lines 8–13). Then, the new sentence pair $(x, y)$ is used to update the SMT model (line 14), and the human-supervised translation is returned (line 15). Otherwise, we directly return the automatic translation $\hat{y}$ as the final translation (line 17).

<table>
<thead>
<tr>
<th>Algorithm 1. Pseudo-code of the proposed cost-sensitive AL framework for CAT. Functions $\text{translate}(M, x)$, $\text{validPrefix}(y)$, $\text{genSuffix}(M, x, y_p)$, and $\text{validTranslation}(y)$ (Section 3.2) denote the IMT user-machine interaction protocol, see Fig. 2. Function $\text{validPrefix}(y)$, $\text{genSuffix}(M, x, y_p)$, and $\text{validTranslation}(y)$ (Section 3.2) denote the IMT user-machine interaction protocol, see Fig. 2. Function $\text{sampling}(B, \rho)$ implements the strategy to sample the most “informative” sentences from $B$ (Section 3.3), and function $\text{update}(M, (x, y))$ returns translation model $M$ updated with the new sentence pair $(x, y)$ (Section 3.4).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>input:</strong> $\mathcal{D}$ (stream of source sentences) $M$ (initial SMT model) $\rho$ (effort level, percentage of sentences to be supervised) $\mathcal{S}$ (block of consecutive sentences) $\mathcal{B}$ (list of sentences to be supervised by the user)</td>
</tr>
<tr>
<td><strong>begin</strong></td>
</tr>
<tr>
<td>repeat</td>
</tr>
<tr>
<td>1 $\mathcal{B} = \text{getBlockFromStream}($ $\mathcal{D}$ $)$</td>
</tr>
<tr>
<td>2 $\mathcal{S} = \text{sampling}($ $\mathcal{B}$ $, \rho)$</td>
</tr>
<tr>
<td>3 for each $x \in \mathcal{B}$ do</td>
</tr>
<tr>
<td>4 $y = \text{translate}(M, x)$</td>
</tr>
<tr>
<td>5 if $x \in \mathcal{S}$ then</td>
</tr>
<tr>
<td>6 $y = \hat{y}$</td>
</tr>
<tr>
<td>7 repeat</td>
</tr>
<tr>
<td>8 $y_p = \text{validPrefix}(y)$</td>
</tr>
<tr>
<td>9 $y_i = \text{genSuffix}(M, x, y_p)$</td>
</tr>
<tr>
<td>10 $y = y_i y_p$</td>
</tr>
<tr>
<td>11 until $\text{validTranslation}(y)$</td>
</tr>
<tr>
<td>12 $M = \text{update}(M, (x, y))$</td>
</tr>
<tr>
<td>13 output(y)</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>14 output(y)</td>
</tr>
<tr>
<td>15 until $\mathcal{D} \neq 0$</td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
</tbody>
</table>

In contrast to previous applications of AL to structured prediction tasks, e.g. sequence labeling (Settles and Craven, 2008), natural language parsing and information extraction (Thompson et al., 1999), or machine translation (Haffari et al., 2009), that minimize the number of labeled samples required to train an accurate model, our goal is to reduce the user supervision effort required to generate high-quality translations. Clearly, the amount of work required to supervise a translation will vary between sentences, e.g. based on the size and the complexity of the source sentence. Thus, it is desirable to design an AL supervision scenario that considers not only how many translations the user is required to supervise, but also how difficult each translation is to supervise.

### 3.2. Translation work-flow and supervision protocol

The proposed AL framework for CAT implies a modification of the conventional IMT work-flow depicted in Fig. 1. The user no longer supervises the translation of all sentences but only of those selected as “worthy of being supervised”. Since only the most informative sentences are supervised, we maximize the utility of each user interaction. Final translations however may not be error-free as for conventional IMT. In exchange, an important reduction in human effort is potentially achievable. Moreover, we can modify the ratio of sentences to be supervised by the user thus modifying the behavior of our system between an automatic SMT system, and a fully-supervised IMT system. In other words, we can adapt the system’s behavior to the requirements of each particular translation task.

Conventional IMT technology is built over the implicit assumption that the inbound text to be translated behaves as a text stream (see Fig. 1). Source sentences are translated separately and no information is stored (or assumed) about the preceding (or following) sentences, e.g. how many sentences remain untranslated. Since the IMT framework uses static SMT models and requires the user to supervise all translations, this is not a strong assumption. However, we have to take it into account because information about previously supervised translations, and particularly, about following sentences may have great impact on the final user effort.

We handle the inbound text stream by partitioning the data into blocks of consecutive sentences. Within a block, all sentences are available, but once the algorithm moves to the next block, all sentences in previous blocks become inaccessible. We use the sentences within a block to estimate the current distribution of sentences in the stream, so that the estimation of the “informativeness” of supervising the translation of a sentence can be done as accurately as possible.

Algorithm 1 shows the pseudo-code that implements the proposed cost-sensitive AL scenario for CAT. The algorithm takes as input a stream of source sentences $\mathcal{D}$, a “base” SMT model $M$, and an effort level $\rho$ denoting the percentage of sentences of each block to be supervised. First, the next block of sentences $\mathcal{B}$ is read from the data stream (line 3). For each sentence in $\mathcal{B}$, the current SMT model generates an initial translation, $\hat{y}$ (line 6). If the sentence has been sampled as worth of supervision, the user collaborates with the system to translate the sentence (lines 8–13). Then, the new sentence pair $(x, y)$ is used to update the SMT model $M$ (line 14), and the human-supervised translation is returned (line 15). Otherwise, we directly return the automatic translation $\hat{y}$ as the final translation (line 17). Although both automatic and user-supervised translations are available, preliminary experiments showed that using both translations to update the SMT model resulted in reduced learning rates.

Although other translation supervision methods, e.g. post-translation, can be used, we implement the IMT user-machine interaction protocol (Fig. 2) to supervise each individual translation. Functions between lines 8–13 denote this supervision procedure:

- $\text{translate}(M, x)$: It returns the most probable automatic translation of $x$ according to $M$. If $M$ is a log-linear SMT model, this function implements Eq. (2.2).
- $\text{validPrefix}(y)$: It denotes the user actions (positioning and correction of the first error) performed to amend $y$. It returns the user-validated prefix $y_p$ of translation $y$, including the user correction $k$.
- $\text{genSuffix}(M, x, y_p)$: It returns the suffix $y_s$ of maximum probability that extends prefix $y_p$. This function implements Eq. (2.4).
- $\text{validTranslation}(y)$: It denotes the user decision of whether translation $y$ is a correct translation or not. It returns True if the user considers $y$ to be correct and False otherwise.
In addition to the supervision procedure, the two elements that define the performance of Algorithm 1 are the sampling strategy $\text{sampling}(B, \rho)$ and the SMT model update function $\text{update}(M, (x, y))$. The sampling strategy decides which sentences of $B$ are worthy of being supervised by the user. This is the main component of our framework and has a major impact on the final performance of the algorithm. Section 3.3 describes several strategies implemented to measure each sentence’s “informativeness.” In turn, the $\text{update}(M, (x, y))$ function updates the SMT model $M$ with a new training pair $(x, y)$. Section 3.4 describes the implementation of this functionality.

3.3. Sentence sampling strategies

The goal of our AL framework for CAT is to generate high-quality translations as effortlessly as possible. Since good translations are less costly to supervise than bad ones, the aim of a sampling strategy $\text{sampling}(B, \rho)$ should be to select those sentences $S \subseteq B$ for which knowing their correct translation allows to improve most the performance of the SMT model for future sentences. To do that, we first use a ranking function $\Phi(x)$ to score the sentences in $B$. Then, the percentage $\rho$ of the highest-scoring sentences are selected to be supervised by the user. We identify three properties that (partially) account for the “worth” of a given sentence:

- Uncertainty: A sentence is as worthy as uncertain is the SMT model of how to translate it.
- Representativeness: A sentence is as worthy as it is “representative” of the sentences in $B$.
- Unreliability: A sentence is as worthy as the amount of unreliably modeled events that it contains.

Next sections describe different sampling strategies designed to measure one (or more) of these complementary properties.

3.3.1. Random ranking ($R$)

Random ranking assigns a random score in the range $[0, 1]$ to each sentence. It is the baseline ranking function used in the experimentation. Although simple, random ranking performs surprisingly well in practice. Its success stems from the fact that it always selects sentences according to the underlying distribution. Using a typical AL heuristic, as training proceeds and sentences are sampled, the training set quickly diverges from the real data distribution. This difficulty known as sampling bias (Dasgupta and Hsu, 2008) is the fundamental characteristic that separates AL from other learning methods. However, since by definition random ranking selects sentences according to the underlying distribution, it does not suffer from sampling bias. This fact makes random ranking a very strong baseline to compare with.

3.3.2. Uncertainty ranking ($U$)

One of the most common AL methods is uncertainty sampling (Lewis and Gale, 1994). This method selects those samples about which the model is least certain how to label. The intuition is clear: much can be learned from the correct output if the model is uncertain of how to label the sample. Formally, a typical uncertainty sampling strategy scores each sample $x$ with one minus the probability of its most probable prediction $y = \text{argmax}_y P(y|x)$:

$$\Phi(x) = 1 - P(y|x)$$  \hspace{1cm} (3.1)

However, due to the peculiarities of SMT models, uncertainty sampling has to be re-considered. Since the normalization term does not influence the decision on the highest-probability translation, it is usually ignored in the model formulation, see Eq. (2.2). As a result, instead of true probabilities these models generate simple scores that are not directly comparable between translations. Hence the conventional uncertainty technique cannot be implemented. Instead, under the assumption that the “certainty” of a model in a particular translation is correlated with the quality of that translation, we measure the uncertainty of a translation using an estimation of its quality. Specifically, we use confidence measures (Blatz et al., 2004; Ueffing and Ney, 2007) to estimate the quality of a translation from the confidence estimations of its individual words.

Given a translation $y = y_1, \ldots, y_n$ generated from a source sentence $x = x_1, \ldots, x_n$, the confidence of each target language word $C(y, x)$ is computed as described in (Ueffing and Ney, 2005):

$$C(y, x) = \max_{0 \leq j \leq |x|} P(y_j|x_j)$$  \hspace{1cm} (3.2)

where $P(y_j|x_j)$ is a word-to-word probability model, and $x_0$ is the empty source word. Following Ueffing and Ney (2005), we use an SMT model 1 (Brown et al., 1993) although other bilingual lexicon models, e.g., model 2 (Brown et al., 1993), or hidden Markov model (Ueffing and Ney, 2007), could also be used.

The confidence-based uncertainty score is then computed as one minus the ratio of words in the most probable translation $y = y_1, \ldots, y_n$ classified as incorrect according to a word-confidence threshold $\tau_w$:

$$\Phi_U(x) = 1 - \frac{|\{y_j : C(y_j, x_j) > \tau_w\}|}{|y|}$$  \hspace{1cm} (3.3)

In the experimentation, threshold value $\tau_w$ was tuned to minimize classification error in a separate development set. Additionally, we use the incremental version of the EM algorithm (Neal and Hinton, 1999) to update the word-to-word probability model $P(y_j|x_j)$ each time a new sentence pair is available.

3.3.3. Information density ranking (ID)

Uncertainty sampling bases its decisions on individual instances which makes the technique prone to sample outliers. The least certain sentences may not be “representative” of other sentences in the distribution, in this case, knowing its label is unlikely to improve accuracy on the data as a whole (Roy and McCallum, 2001). We can overcome this problem by modeling the input distribution explicitly when scoring a sentence.

The information density framework (Settles and Craven, 2008) is a general density-weighting technique. The main idea is that informative instances should not only be those which are uncertain, but also those which are “representative” of the underlying distribution (i.e., inhabit dense regions of the input space). To address this, we compute the information density score:

$$\Phi_{ID}(x) = \Phi_U(x) \cdot \left( \frac{1}{|B|} \sum_{b=1}^{|B|} S(x, x_b) \right)^7$$  \hspace{1cm} (3.4)

where the uncertainty of a given sentence $x$ is weighted by its average similarity $S(x, \cdot)$ to the rest of sentences in the distribution, subject to a parameter $\gamma$ that controls the relative importance of the similarity term. Since the distribution is unknown, we use the block of sentences $B = \{x_1, \ldots, x_n\}$ to approximate it. We use uncertainty ranking $\Phi_U(x)$ to measure the “base” worth of a sentence, but we could use any other instance-level strategies.

---

3 We use the same symbol $|\cdot|$ to denote an absolute value $|a|$, the length of a sequence $|x|$, and the cardinality of a set $|\mathcal{B}|$. The particular meaning will be clear depending on the context.
presented in the literature (Settles and Craven, 2008; Haffari et al., 2009).

We compute the similarity of two sentences as the geometric mean of the precision of n-grams (sequences of n consecutive words in a sentence) up to size four \( ^4 \) between them:

\[
S(\mathbf{x}, \mathbf{x}_b) = \left( \prod_{n=1}^{4} \frac{\sum_{w=1}^{W_n} \min(\#_w(\mathbf{x}), \#_w(\mathbf{x}_b))}{\sum_{w=1}^{W_n} \#_w(\mathbf{x})} \right)^{\frac{1}{4}}
\]

(3.5)

where \( W_n(x) \) is the set of n-grams of size \( n \) in \( \mathbf{x} \), and \( \#_w(\mathbf{x}) \) represents the count of n-gram \( w \) in \( \mathbf{x} \). This similarity score is closely related to the widespread translation evaluation score BLEU (Papineni et al., 2002) that will be further discussed in Section 4.2.1.

One potential drawback of information density is that the number of similarity calculations grows quadratically with the number of instances in \( B \). However, similarities only need to be computed once for a given \( B \) and are independent of the base measure. Therefore, we can pre-compute and cache them for efficient look-up during the AL process.

3.3.4. Coverage augmentation ranking (CA)

Sparse data problems are ubiquitous in natural language processing (Ziff, 1935). This implies that some rare events will be missing completely from a training set, even when it is very large. Missing events result in a loss of coverage, a situation when the structure of the model is not rich enough to cover all types of input. As a result, words (or sequences thereof) that do not appear in the training set cannot be adequately translated (Turchi et al., 2009; Haddow and Koehn, 2012).

Uncertainty sampling assumes that the model structure is fixed in advance and focus upon improving parameters within that structure. However, this is not appropriate for SMT where the model structure and the associated parameters are determined from training data. The problem is that uncertainty-based methods fail at dealing with sentences with words not covered by the model. To efficiently reduce classification error in SMT, we should explicitly address unreliably trained model parameters. We do that by measuring the coverage augmentation \( \Delta_{cov}(x, T) \) due to the incorporation of sentence \( x \) to the current training set \( T \):

\[
\Delta_{cov}(x, T) = \sum_{n=1}^{4} \sum_{w=1}^{W_n(x)} \#_w(x)
\]

(3.6)

The coverage augmentation for each sentence \( x \) is given by the count of n-grams in \( x \) missing in the training set \( T \) that appear in the rest of sentences in the block. That is, we measure how many missing n-grams in \( B \) would be covered if \( x \) is added to the training set. Again, we consider \( n = 4 \) as the maximum n-gram length.

This coverage augmentation score is biased towards longer sentences since longer sentences can contain a larger amount of unseen n-grams. This is one of the reasons for its successful application in conventional AL scenarios (Haffari et al., 2009) and bilingual sentence selection tasks (Gascó et al., 2012). However, longer sentences also imply a higher cognitive effort from the user (Koponen, 2012) which may penalize performance. We address this dilemma by normalizing the coverage augmentation score by an estimation of the user-effort \( E(x) \) required to supervise the translation. Since out-of-coverage words cannot be adequately translated and their translations will be corrected by the user, we assume user effort to be proportional to the number of out-coverge-words in the source sentence:

\[
E(x) \propto \sum_{w \in (W_n(x) - \hat{W}_n(T))} \#_w(x)
\]

(3.7)

Finally, the coverage augmentation score measures the potential SMT model improvement per unit of user effort:

\[
\Phi_{cov}(x, T) = \frac{\Delta_{cov}(x, T)}{E(x)}
\]

(3.8)

To avoid selecting several sentences with the same missing n-grams, we update the set of n-grams seen in training each time a new sentence is selected. First, sentences in \( B \) are scored using Eq. (3.8). Then, the highest-scoring sentence is selected and removed from \( B \). The set of training n-grams is updated with the n-grams present in the selected sentence and, hence, the scores of the rest of the sentences in the block are also updated. This process is repeated until we select the desired ratio \( \rho \) of sentences from \( B \).

3.4. Online training for SMT

After the translation supervision process, we have a new sentence pair \((x, y)\) at our disposal. We now briefly describe the incremental SMT model used in the experimentation, and the online learning techniques implemented to update the model with new sentence pairs in constant time.

We implement the online learning techniques proposed in Ortiz-Martínez et al. (2010). In that work, a state-of-the-art log-linear SMT model (Och and Ney, 2002) was presented. This model is composed of a set of incremental feature functions governing different aspects of the translation process, see Eq. (2.2), including a language model, a model of source sentences length, direct \( P(y|x) \) and inverse \( P(x|y) \) phrase-based \(^6\) translation models (Koehn et al., 2003), models of the length of the source and target language phrases, and a reordering model.

Together with this log-linear SMT model, Ortiz-Martínez et al. (2010) present online learning techniques that, given a training pair, update the incremental features. In contrast to conventional batch learning techniques, the computational complexity of adding a new training pair is constant, i.e., it does not depend on the number of training samples. To do that, a set of sufficient statistics is maintained for each feature function. If the estimation of the feature function does not require the use of the EM algorithm (Dempster et al., 1977) then it is generally easy to incrementally update the feature given the new training sample. For example, to update a language model with the new translation we simply have to update the current count of each n-gram in \( y \). By contrast, if the EM algorithm is required (e.g. to estimate phrase-based SMT models) the estimation procedure has to be modified because EM is designed to be used in batch learning scenarios. For such feature functions, the incremental version of the EM algorithm (Neal and Hinton, 1999) is applied. For example, phrase-based models are estimated from an hidden Markov (HMM) model (Ueffing and Ney, 2007). Since the HMM model is determined by a hidden alignment variable, the incremental version of the EM algorithm is required to update the model with the new training sample \((x, y)\).

A detailed description of the update algorithm for each feature function was presented in Ortiz-Martínez et al. (2010).

\(^4\) Papineni et al. (2002) obtained the best correlation with human judgments using n-grams of maximum size \( n = 4 \).

\(^5\) We ignore the effort proportionality constant since it is equal for all sentences.

\(^6\) In contrast with word-based translation models where the fundamental translation unit is the word, phrase-based models translate whole sequences of words. These sequences are called phrases although typically they are not linguistically motivated.
4. Experiments

We carried out experiments to assess the performance of the proposed cost-sensitive AL framework for CAT. The idea is to simulate a real-world scenario where a translation agency is hired to translate a huge amount of text. The experimentation was divided into two parts. First, Section 4.3 describes a typical AL experimentation, such as the one in Haffari et al. (2009), where we studied the learning curves of the SMT model as a function of the number of training sentence pairs. Then, Section 4.4 focuses on the productivity of the whole CAT system. There, we measured, for each ranking function, the quality of the final translations generated by the system as a function of the supervision effort required from the user. With this experimentation, we can observe how the improvements of the underlying SMT model are reflected in the productivity of the whole cost-sensitive AL CAT system.

4.1. Methodology and data

The experimentation carried out comprises the translation of a test corpus using different setups of the proposed cost-sensitive AL framework. Each setup was defined by the ranking function used. All experiments start with a “base” SMT model whose feature functions are trained on the training partition of the Europarl (Koehn and Monz, 2006) corpus, and its log-linear weights are tuned by minimum error-rate training (Och, 2003) to optimize BLEU (Papineni et al., 2002) in the development partition. Then, we run Algorithm 1 until all sentences of the News Commentary corpus (Callison-Burch et al., 2007) are translated into English. We use blocks of size \( |B| = 1000 \) (González-Rubio et al., 2012) show that similar results were obtained with other block sizes), and for information density, we arbitrarily set \( \gamma = 1 \) (i.e., uncertainty and density terms had equal importance). The main figures of the training, development, and test corpora are shown in Table 1.

The reasons to choose the News Commentary corpus as test corpus are threefold: its size is large enough to test the proposed techniques in the long term, its sentences come from a different domain (news) than the sentences in the Europarl corpus (proceedings of the European parliament), and it contains sentences of different topics which allows us to test the robustness of our system against topic-changing data streams. Therefore, by translating the News Commentary corpus we simulate a realistic scenario where translation agencies must be ready to fulfill eclectic real-world translation requirements.

Since an evaluation involving human users is too expensive, we use the reference translations of the News Commentary corpus to simulate the target translations which a human user would want to obtain. At each interaction (see Fig. 2), the prefix validated by the user is computed as the longest common prefix between the translation suggested by the system \( \mathbf{y} \) and the reference translation \( \mathbf{r} \), and the user correction \( (k) \) is given by the first mismatched character between \( \mathbf{y} \) and \( \mathbf{r} \). The interaction continued until the longest common prefix is equal to the reference translation.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Use</th>
<th>Sentences</th>
<th>Tokens</th>
<th>Vocabulary</th>
<th>Out-of-coverage tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(Spa/Eng)</td>
<td>(Spa/Eng)</td>
<td></td>
<td>(Spa/Eng)</td>
</tr>
<tr>
<td>Europarl Training</td>
<td>731k</td>
<td>15 7M/</td>
<td>103k/64k</td>
<td>–/–</td>
<td></td>
</tr>
<tr>
<td>Europarl Development</td>
<td>2k</td>
<td>60k/58k</td>
<td>7k/6k</td>
<td>208/127</td>
<td></td>
</tr>
<tr>
<td>News Test</td>
<td>51k</td>
<td>1.5M/5k</td>
<td>48k/35k</td>
<td>133/11k</td>
<td></td>
</tr>
<tr>
<td>News commentary</td>
<td>1.2M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.2. Evaluation measures

The goal of the proposed cost-sensitive AL framework is to obtain high translation quality with as few user effort as possible. Therefore, the evaluation is twofold: quality of the generated translations and amount of supervision effort required to generate them. Additionally, we describe how we compute the statistical significance of the results.

4.2.1. Measuring translation quality

We evaluate translation quality using the well-established BLEU (Papineni et al., 2002) score. BLEU computes the geometric mean of the precision of \( n \)-grams of various lengths between a candidate translation and a reference translation. This geometric average is multiplied by a factor, namely the brevity penalty, that penalizes candidates shorter than the reference. Following the standard implementation, we consider \( n = 4 \) as the maximum \( n \)-gram length. BLEU is a percentage that measures to which extent the candidate translation candidate translation contains the same information as the reference translation. Thus, a BLEU value of 100% denotes a perfect match between the candidate translation and the reference translation.

4.2.2. Measuring supervision effort

We estimate the user effort as the number of user actions required to supervise a translation which depend on the supervision method.\(^7\) In the interaction protocol described in Section 3.2, the user can perform two different actions to interact with the system. The first action corresponds to the user looking for the next error and moving the pointer to the corresponding position of the translation hypothesis. The second action corresponds to the user replacing the first erroneous character with a keystroke.

To do that, we compute the keystroke and mouse-action ratio (KSMR) (Barrachina et al., 2009) which has been extensively used to report user effort results in the IMT literature. KSMR is calculated as the number of keystrokes plus the number of movements (mouse actions) divided by the total number of characters of the reference translation. From a user point of view the two types of actions are different, and may require different types of effort (Macklovitch, 2006). A weighted measure could take this into account; however, in these experiments, we assume each action has unit cost.

4.2.3. Statistical significance

We apply statistical significance testing to establish that an observed performance difference between two methods is in fact significant, and has not just arisen by chance. We state a null hypothesis: “Methods A and B do not differ with respect to the evaluation measure of interest” and determine the probability, namely the \( p \)-value, that an observed difference has arisen by chance given the null hypothesis. If the \( p \)-value is lower than a certain significance level (usually \( p < 0.01 \), or \( p < 0.05 \)) we can reject the null hypothesis. To do that, we use randomization tests because they free us from worrying about parametric assumptions and they are no less powerful than ordinary t-tests (Noreen, 1989).

Specifically, we use a randomization version of the paired t-test based on (Chinchor, 1992):

1. Collect the absolute difference in evaluation measure \( Q(\cdot) \) for methods A and B
   \[ Q(A) - Q(B) \]

\( ^7 \) For example, if instead of using the IMT supervision protocol we ask the user to post-edit the translations, user actions are edit operations, and the natural effort measure is the word error rate, also known as Levenshtein distance.
2. Shuffle $N$ times ($N = 999$ in our experiments).
3. Count the number of times ($N_p$) that $|Q(A) - Q(B)| > |Q(A') - Q(B')|$.
4. The estimate of the $p$-value is $\frac{N_p + 1}{N}$ (1 is added to achieve an unbiased estimate).

Initially, we use an evaluation measure $Q()$ (e.g. BLEU) to determine the absolute difference between the original outcomes of methods $A$ and $B$. Then, we repeatedly create shuffled versions $A'$ and $B'$ of the original outcomes, determine the absolute difference between their evaluation metrics, and count the number of times $N^p$ that this difference is equal or larger than the original difference. To create the shuffled versions of the data sets, we iterate over each data point in the original outcomes and decide based on a simulated coin-flip whether data points should be exchanged between $A$ and $B$. The $p$-value is the proportion of iterations in which the absolute difference in evaluation metric was indeed larger for the shuffled version (corrected to achieve an unbiased estimate).

4.3. Active learning results

We first studied the learning rates of the different ranking functions in a typical AL experimentation. Here, the performance of the SMT model is studied as a function of the percentage $\rho$ of the corpus used to update it. SMT model performance was measured as the translation quality (BLEU) of the initial automatic translations measuring user supervision effort. We studied the user effort required to supervise translations as a function of the percentage of sentences $\rho$ supervised. Fig. 4 shows the KSMR scores obtained by each ranking function, and the significance level of some pairwise ranking function comparisons.

Results show that sentences selected by coverage augmentation consistently outperformed the random ranking baseline. Additionally, the observed difference was statistically significant as shown in the second panel of the figure. This result shows that coverage augmentation is the ranking function that more effectively detected those sentences that improve most the performance of the SMT model.

Both uncertainty ranking and information density ranking were outperformed by random ranking when supervising up to 50% of the corpus; after that, results for the three ranking functions were very similar and almost no statistical difference was observed (third panel). Additionally, uncertainty ranking and information density ranking obtained virtually the same results; however the slightly better results of uncertainty ranking were statistically significant (fourth panel). That is, the addition of the “representativeness” in information density deteriorated the performance of uncertainty ranking. This counter-intuitive result can be explained by the intrinsic sparse nature of natural language, and particularly by the eclectic topics, e.g. economic, science, or politics, of the sentences in the test corpus.

In the previous experiment, we assumed that all translations were equally costly to supervise. However, different sentences involve different translation costs. Therefore, we then focused on measuring user supervision effort. We studied the user effort required to supervise translations as a function of the percentage of sentences $\rho$ supervised. Fig. 4 shows the KSMR scores obtained by each ranking function, and the significance level of some pairwise ranking function comparisons.

Results show that sentences selected by coverage augmentation required a statistically significant larger amount of effort than the ones selected by random; except when supervising almost all
sentences $\rho > 96\%$ where coverage augmentation required a lower amount of effort (second panel in Fig. 4). This indicates that even when all sentences are supervised $\rho = 100\%$ the order in which they are supervised (depending on the ranking function) affects the efficiency of the supervision process.

Regarding uncertainty and information density, both ranking functions required a statistically lower amount of effort than random (third panel), and similarly to the results in Fig. 3, differences between uncertainty and information density were scarce but statistically significant (fourth panel). In this case, sentences selected by information density required a statistically lower amount of effort to be supervised.

### 4.4. Productivity results

Results in the previous section show that those ranking functions that obtained better learning rates are also those that required more supervision effort, and vice versa. However, from a point of view of a translation agency that has to invest its limited resources, the key point is how to obtain the better productivity. That is, given a required translation quality, how to reduce supervision effort; or symmetrically, given an effort level, how to maximize translation quality.

To answer these questions, we studied the relation between user effort and final translation quality. In contrast with the experimentation in Fig. 3 where we study the learning rates of the SMT model by measuring the quality of its automatic translations, we now are interested in the performance of the complete cost-sensitive AL system. We did that by measuring the translation quality of the translations generated by Algorithm 1 (lines 15 and 17) as a function of the required supervision effort. Note that this final translations are a mixture of automatic and user-supervised translations. The ratio between them is fixed by $\rho$ which permits to adjust system’s behavior between a fully automatic SMT system if none translation is supervised ($\rho = 0\%$), or a conventional IMT system where all translations are supervised ($\rho = 100\%$).

Since uncertainty and information density obtain so similar performance in the previous experiments, Fig. 5 compares the performance of only random (R), information density ranking (ID), and coverage augmentation (CA) ranking functions. Additionally, we present results of the proposed cost-sensitive AL framework using a static SMT model. The objective was to test the influence of SMT model updating on translation productivity.

Results show a huge leap in productivity when the SMT model was updated with user feedback. This continuous model updating allowed to obtain twice the translation quality with the same level of supervision effort. Regarding the different ranking functions, both information density and coverage augmentation performed similarly yielding slight improvements in productivity with
respect to random, particularly for high levels of effort. For example, if a translation quality of 60% BLEU is acceptable, then the human translator would need to modify only a 20% of the characters of the automatically generated translations.

5. Conclusions and future work

We have presented a cost-sensitive AL framework for CAT designed to boost translation productivity. The two cornerstones of our approach are the selective supervision protocol and the continual SMT model updating with user-supervised translations. Regarding selective supervision, we propose to focus user effort on a subset of sentences that are considered "worth of being supervised" according to a ranking function. The percentage of sentences to be supervised is defined by a tunable parameter which allows to adapt the system to meet task requirements in terms of translation quality, or resources availability. Whenever a new user-supervised translation pair is available, we use it to update a log-linear model. Different online learning techniques are implemented to incrementally update the model.

We evaluated the proposed cost-sensitive AL framework in a simulated translation of real data. Results showed that the use of user-supervised translations reduced to one half the effort required to translate the data. Additionally, the use of an adequate ranking function further improved translation productivity.

The experimental simulation carried out is effective for evaluation, but, to assess the obtained results, we plan to conduct a complete study involving real human users. Productivity could be measured by the actual time it takes a user to translate a test document. This evaluation additionally requires addressing issues of user interface design and user variability, but it is ultimately the most direct evaluation procedure.

An additional direction for further research is to study why random ranking performs so well. We have provided some insights of which are the reasons for this, but we hope that a further study will reveal new hints that may guide us towards the definition of sampling strategies that outperform random sampling. Moreover, the study of productivity-focused ranking functions is a wide research field that should also be explored.

Acknowledgments

Work supported by the European Union Seventh Framework Program (FP7/2007-2013) under the CasMaCat Project (Grants agreement No. 287576), by the Generalitat Valenciana under Grant ALMPR (Prometeo/2009/014), and by the Spanish Government under Grant TIN2012-31723. The authors thank Daniel Ortiz-Martínez for providing us with the log-linear SMT model with incremental features and the corresponding online learning algorithms. The authors also thank the anonymous reviewers for their criticisms and suggestions.

References


