D2.1: Initial Report on Interactive Translation Prediction

Germán Sanchis-Trilles, Vicent Alabau, Francisco Casacuberta, José-Miguel Benedí, Jorge González, Alfons Juan-Ciscar, Daniel Ortiz-Martínez, Philipp Koehn

Distribution: Public

CASMACAT
Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation

ICT Project 287576 Deliverable D2.1

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

<table>
<thead>
<tr>
<th><strong>Distribution</strong></th>
<th><strong>Public</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Contractual date of delivery</strong></td>
<td>October 31, 2012</td>
</tr>
<tr>
<td><strong>Actual date of delivery</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Date of last update</strong></td>
<td>July 16, 2013</td>
</tr>
<tr>
<td><strong>Deliverable number</strong></td>
<td>D2.1</td>
</tr>
<tr>
<td><strong>Deliverable title</strong></td>
<td>Initial Report on Interactive Translation Prediction Report</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Draft</td>
</tr>
<tr>
<td><strong>Status &amp; version</strong></td>
<td>Draft</td>
</tr>
<tr>
<td><strong>Number of pages</strong></td>
<td>71</td>
</tr>
<tr>
<td><strong>Contributing WP(s)</strong></td>
<td>WP2</td>
</tr>
<tr>
<td><strong>WP / Task responsible</strong></td>
<td>UPVLC</td>
</tr>
<tr>
<td><strong>Other contributors</strong></td>
<td>CS</td>
</tr>
<tr>
<td><strong>Internal reviewer</strong></td>
<td>Philipp Koehn</td>
</tr>
<tr>
<td><strong>Author(s)</strong></td>
<td>Germán Sanchis-Trilles, Vicent Alabau, Francisco Casacuberta, José-Miguel Benedí, Jorge González, Alfons Juan-Ciscar, Daniel Ortiz-Martínez, Philipp Koehn</td>
</tr>
<tr>
<td><strong>EC project officer</strong></td>
<td>Kimo Rossi</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.
1 Executive Summary

This deliverable details the progress done in the first year of WP2, Interactive Translation Prediction, of the CASMACAT project. This work-package has been one of the most active ones during the first year, with four of its five tasks already active. Since this WP focuses on the basic aspects of translation prediction, the work performed implies in some cases basic research, including the development and implementation of completely novel SMT models, new search strategies, or proposing different theoretical frameworks for the prediction of the user’s interaction. The main goal of this work-package is to develop novel strategies and algorithms that provide flexibility and efficiency to the interaction process, be it by means of more adequate machine translation systems and prediction algorithms, or by allowing the introduction of multimodality into the conventional ITP interface.

During this first year, work was performed in tasks 2.1, 2.2, 2.4 and 2.5:

Task 2.1 Search and Machine Learning Criteria for Prediction (month 1-18)
The work performed in this task concerned mainly two aspects of the prediction mechanism. On the first place, we reviewed the traditional approach to the prediction problem, which relies typically on a maximum-a-posteriori approach, and proposed an alternative approach, motivated by the Bayesian decision theory. Work on this field has already yielded satisfactory results, supported by a publication in an international journal, but is still ongoing, and currently a method is being studied to transform the optimum prediction algorithm using word-graphs into a greedy version. On the second place, a new error-correcting strategy was developed, so that the system is not so heavily penalised whenever the user intends to produce a translation which the machine translation system is not able to account for. Work in this direction is reporting very promising results, and will soon be submitted for publication to an international journal. Further work to be conducted in the rest of the period scheduled for this task involves including IBM models into the error-correcting procedure.

Task 2.2 Multi-modality in Interactive Translation Prediction (month 6-24)
Work in this direction has focused so far in using an e-pen for interacting with the post-editing or ITP system, both for allowing the expert translator to correct the hypotheses by just writing on a screen or for allowing e-pen gestures. The work conducted until now has lead to a publication in an international conference, but there is still much work to be done. For instance, we intend to extend the interaction interface by allowing multi-word corrections, since only single-word corrections are allowed currently. In addition, the e-pen gesture recognition system has only been simulated so far, but still needs to be implemented into the CASMACAT workbench.

Task 2.4 Prediction from Parse Forest (month 6-18) The work conducted in the first months of this task concerns the development and implementation of a toolkit for producing parse-forests from an ITG translation engine. A toolkit able to produce translations was already available, and was been published in an international conference. Such toolkit has been extended in order to produce parse forests, and is now being complemented for its use within an interactive setting.

Task 2.5 New SMT Models for Interactive Translation Prediction (month 1-12)
The purpose of this task was to study different novel SMT models for their application within the interactive framework. In this task, we investigated two different SMT models, namely a phrase-based hidden semi-Markov model and a phrase-based finite state model. These models were able to yield very satisfactory results in constrained tasks, and were published in different journals and conferences. However, they were not able to perform better than a fully-fledged SMT system trained by Moses in a more generic setting, be it because of monotonicity or because of sentence length and temporal restrictions. For this reason, the results obtained only report translation quality in a conventional SMT setting, since these models were finally not applied within an interactive framework.
Contents

1 Executive Summary

2 Background
   2.1 Statistical Machine Translation ........................................... 5
   2.2 Interactive Machine Translation ............................................. 6

3 Task 1: Search and machine learning criteria for prediction
   3.1 Optimal decision rule for sequential interactive structured prediction .................. 7
      3.1.1 Introduction ................................................................. 7
      3.1.2 Development .................................................................. 7
      3.1.3 Conclusions ................................................................. 7
   3.2 IMT Based on Stochastic Error-Correction Models .............................................. 8
      3.2.1 PFSMs as Stochastic Error-Correction Models ........................................ 8
      3.2.2 Alternative IMT Formalisation ............................................ 11
      3.2.3 Experiments .................................................................. 14
      3.2.4 Experiments with the EU Corpus ......................................... 16
      3.2.5 Conclusions ................................................................. 17

4 Task 2: Multi-modality in interactive translation prediction ................................. 17
   4.1 Introduction ........................................................................ 17
   4.2 Robust on-line HTR by leveraging MT models ............................................. 18
   4.3 Study of e-pen gestures for ITP ..................................................... 18
   4.4 Conclusions ....................................................................... 18

5 Task 4: Prediction from Parse Forest ....................................................................... 19
   5.1 Introduction ........................................................................ 19
   5.2 Exact Search for Partial Translation .................................................. 19
      5.2.1 Model ........................................................................ 19
      5.2.2 Algorithms for Interactive Translation From Parse Forests ...................... 20
      5.2.3 Conclusions ................................................................. 21
   5.3 Approximate Search for User Prefix ..................................................... 22
      5.3.1 Formal Definition .............................................................. 22
      5.3.2 Example ...................................................................... 23
      5.3.3 Algorithm .................................................................... 24
      5.3.4 Refinements ................................................................. 26

6 Task 5: New SMT Models for ITP ......................................................................... 26
   6.1 Phrase-based hidden semi-Markov models .............................................. 26
      6.1.1 Introduction ................................................................. 26
      6.1.2 The phrase-based hidden semi-Markov model ..................................... 27
      6.1.3 Recurrences ................................................................. 28
      6.1.4 Training ...................................................................... 29
      6.1.5 Experiments ................................................................. 30
      6.1.6 Conclusions ................................................................. 33
   6.2 Finite-state models ..................................................................... 34
      6.2.1 Introduction ................................................................. 34
      6.2.2 Development ................................................................. 34
      6.2.3 Conclusions ................................................................. 35

Bibliography ................................................................. 36

Attachment A ................................................................. 39
2 Background

Despite of the multiple and important advances obtained so far in the field of Statistical Machine Translation (SMT), current machine translation (MT) systems are not able to produce ready-to-use texts \cite{1,11}. Indeed, MT systems usually require human post-editing in order to achieve high-quality translations.

One way of taking advantage of MT systems is to interactively combine them with the knowledge of a human translator, constituting the Interactive Machine Translation (IMT) paradigm. This IMT paradigm can be considered a special type of the so-called computer-assisted translation (CAT) paradigm \cite{17}. The main difference between IMT and traditional post-editing is that, in IMT, the system takes into account each interaction of the user, and attempts to build an improved translation hypothesis by completing the suffix validated by the user. Typical implementations of IMT systems are based on the generation of word translation graphs. During the interactive translation process of a given source sentence, the system makes use of the word graph generated for that sentence in order to complete the prefixes accepted by the human translator. Specifically, the system finds the best path in the word graph which is compatible with the user prefix.

The main advantages of word graph-based IMT systems is their efficiency in terms of the time cost per each interaction. This is due to the fact that the word graph is generated only once at the beginning of the interactive translation process of a given source sentence, and the suffixes required in IMT can be obtained by incrementally processing this word graph.

On the contrary, a common problem in IMT arises when the user sets a prefix which cannot be explained by the statistical models. Under these circumstances, the suffix cannot be appropriately generated, since the system is unable to generate translations that are compatible with the prefix validated by the user. In those IMT systems that use word graphs to generate the suffixes, the common procedure to face this problem is to perform a tolerant search in the word graph. This tolerant search uses the well known concept of Levenshtein distance in order to obtain the most similar string for the given prefix (see \cite{29} for more details). These error-correcting techniques, although they are not included in the statistical formulation of the IMT process, are crucial to ensure that the suffixes completing the prefixes given by the user can be generated.

In this work, an alternative formalisation of the IMT framework which includes stochastic error-correction models in its statistical formulation is proposed. The proposed technique relies on word graphs to generate the suffixes required in IMT.

2.1 Statistical Machine Translation

The statistical approach to MT formalises the problem of generating translations under a statistical point of view. More formally, given a source sentence $x^T_1 ≡ x_1...x_j...x_J$ in the source language $X$, we want to find its equivalent target sentence $y^T_1 ≡ y_1...y_i...y_I$ in the target language $Y$.

$^1x_j$ and $y_i$ note the $i$'th word and the $j$'th word of the sentences $x^T_1$ and $y^T_1$ respectively.
From the set of all possible sentences of the target language, we are interested in that with the highest probability according to the following equation:

\[ \hat{y}_1^I = \arg \max_{I, y_1^I} \{ \Pr(y_1^I \mid x_1^I) \} \]  

(1)

The early works on SMT were based on the use of generative models. A generative model is a full probability model of all statistical variables that are required to randomly generating observable data. Generative models decompose \( \Pr(y_1^I \mid x_1^I) \) applying the Bayes decision rule. Taking into account that \( \Pr(x_1^I) \) does not depend on \( y_1^I \) we arrive to the following expression [10]:

\[ \hat{y}_1^I = \arg \max_{I, y_1^I} \{ \Pr(y_1^I) \cdot \Pr(x_1^I \mid y_1^I) \} \]  

(2)

where: \( \Pr(y_1^I) \) represents the probability of generating the target sentence, and \( \Pr(x_1^I \mid y_1^I) \) is the probability of generating \( y_1^I \) given \( x_1^I \). Since the real probability distributions \( \Pr(y_1^I) \) and \( \Pr(x_1^I \mid y_1^I) \) are not known, they are approximated by means of parametric statistical models. Specifically, \( \Pr(y_1^I) \) is modelled by means of a language model, and \( \Pr(x_1^I \mid y_1^I) \) is modelled by means of a translation model. Current MT systems are based on the use of phrase-based models [21] as translation models. Typically, the values of the parameters of such statistical models are obtained by means of the well-known maximum-likelihood estimation method.

More recently, alternative formalisations have been proposed. Such formalisations are based on the direct modelling of the posterior probability \( \Pr(y_1^I \mid x_1^I) \), replacing the generative models by discriminative models. Log-linear models use a set of feature functions \( h_m(x_1^I, y_1^I) \) each one with its corresponding weight \( \lambda_m \):

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

(3)

The direct optimisation of the posterior probability in the Bayes decision rule is referred to as discriminative training [28]. Since the features of regular SMT log-linear models are usually implemented by means of generative models, discriminative training is applied here only to estimate the weights involved in the log-linear combination. This process is typically carried out by means of the minimum error rate training (MERT) algorithm [32].

2.2 Interactive Machine Translation

As was already mentioned in previous section, the IMT framework constitutes an alternative to fully automatic MT systems in which the MT system and its user collaborate to generate correct translations. These correct translations are generated in a series of interactions between the IMT system and its user. Specifically, at each interaction of the IMT process, the IMT system generates a translation of the source sentence which can be partially or completely accepted and corrected by the user of the IMT system. Each partially corrected text segment, or prefix, is then used by the SMT system as additional information to generate better translation suggestions.

More formally, in the IMT scenario we have to find an extension \( y_s \) for a prefix \( y_p \) given by the user:

\[ \hat{y}_s = \arg \max_{y_s} \{ p(y_s \mid x_1^I, y_p) \} \]  

(4)

Applying the Bayes rule, we arrive at the following expression:

\[ \hat{y}_s = \arg \max_{y_s} \{ p(y_s \mid y_p) \cdot p(x_1^I \mid y_p, y_s) \} \]  

(5)
where the term \( p(y_p) \) has been dropped since it does not depend on \( y_s \).

Thus, the search is restricted to those sentences \( y' \) which contain \( y_p \) as prefix. It is also worth mentioning that the similarities between Equation (6) and Equation (2) (note that \( y_p y_s \equiv y'_1 \)) allow us to use the same models if the search procedures are adequately modified [9, 8].

Note that IMT has not been yet evaluated in the first trial, where the focus was mainly to assess the user interface, but is to be evaluated in the second field trial.

3 Task 1: Search and machine learning criteria for prediction

3.1 Optimal decision rule for sequential interactive structured prediction

3.1.1 Introduction

Traditionally, ITP systems have used a maximum-a-posteriori (MAP) approach to the problem of searching for a good suffix. MAP is known to be an optimal decision rule when we target for a perfect suffix. However, ITP does not aim to produce perfect automatic translations but to allow the user to obtain the desired output with less editing effort.

3.1.2 Development

In [4] (see Attachment A) we derive an optimal decision rule for the suffix search problem that follows the Bayes decision theory. The resulting decision rule works by appending from left-to-right and word-by-word the most like word that continues the prefix. The probability of the next word can be obtained by summing up over all the possible suffixes starting with that word, which can be exponential in number. Next, we prove that MAP can be seen as a Viterbi-like approximation of the optimal approach where the summation over the suffixes is replaced by a maximum. Therefore, it is expected that if the maximum dominates over the summation, then MAP and the optimal approach obtain the same result. Another result is that it is then trivial to prove that MAP is not an admissible decision rule.

Regarding the practical aspects, we have devised an algorithm to make the computation of a large number of the most likely hypothesis. This way, the problem of exponential number of suffixes to sum up can be approached efficiently by using a subset of the hypothesis space, although the results are not exact. As the MT search problem can be approached as a dynamic programming algorithm or as a stack decoding problem with hypothesis recombination, word graph containing large number of hypothesis can be obtained as a sub-product of the search algorithm. The experiments show that, in some cases, the optimal approach obtains statistically significant improvements with respect to MAP. However, MAP usually performs as well as the optimal approach.

3.1.3 Conclusions

The empirical evaluation has shown that the optimal decision rule can be obtained efficiently from word graphs. Although the improvements are not decisive, the optimal algorithm should be used, since it comes at little computational cost with respect to previous approaches. However, MAP can be used safely if the circumstances require it. For the rest of the duration of this task, we intend to study the possibility of transforming the optimum decision rule based on word graphs into a greedy version of the algorithm, which would perform faster and decrease response time. In addition, the experiments performed as of yet are only preliminary, and are reported over corpora which are not the standard corpora present in CASMACAT. Hence, future work also involves obtaining results with such corpora.

---

\(^2^\)An admissible decision rule is a rule for making a decision such that there isn’t any other rule that is always better than it [13].
3.2 IMT Based on Stochastic Error-Correction Models

As has been already mentioned, a common problem in IMT arises when the user sets a prefix which cannot be explained by the statistical models. This problem requires the introduction of specific techniques in the IMT systems to guarantee that the suffixes can be generated. The vast majority of the IMT systems described in the literature (with the exception of the work presented in [42]) apply error-correcting techniques based on the concept of edit distance to solve the coverage problems. These error-correcting techniques, although they are not included in the statistical formulation of the IMT process, are crucial to ensure that the suffixes completing the prefixes given by the user can be generated.

In this section an alternative formalisation of the IMT framework which includes stochastic error-correction models in its statistical formulation is proposed. For this purpose, we adapt probabilistic finite-state machines (PFSMs) for its use as stochastic error-correction models.

3.2.1 PFSMs as Stochastic Error-Correction Models

To the best of our knowledge, the first stochastic interpretation of edit distance was described in [7]. In that work, PFSMs were used to model the transformations produced by a noisy channel in a given text string.

A PFSM (see [43, 44] for a detailed description) is a tuple \( A = \langle Q_A, \Sigma, \delta_A, I_A, F_A, P_A \rangle \), where:

- \( Q_A \) is a finite set of states;
- \( \Sigma \) is the alphabet;
- \( \delta_A \subseteq Q_A \times \Sigma \cup \{ \lambda \} \times Q_A \) is a set of transitions;
- \( I_A : Q_A \rightarrow \mathbb{R}^+ \) (initial-state probabilities);
- \( P_A : \delta_A \rightarrow \mathbb{R}^+ \) (transition probabilities);
- \( F_A : Q_A \rightarrow \mathbb{R}^+ \) (final-state probabilities).

\( I_A, P_A \) and \( F_A \) are functions such that:

\[
\sum_{q \in Q_A} I_A(q) = 1
\]

and

\[
\forall q \in Q_A, F_A(q) + \sum_{a \in \Sigma, q' \in Q_A} P_A(q, a, q') = 1
\]

In what follows, we will use \( q \) without subindex to denote a generic state of \( Q \), the specific states of \( Q \) will be denoted as \( q_0, q_1, ..., q_{|Q|−1} \), and a sequence of states of length \( j \) will be denoted as \( (s_0, s_1, ..., s_j) \) where \( s_i \in Q \) for \( 1 \leq i \leq j \).

PFSMs are stochastic machines that generate probabilities for a set of finite strings contained in \( \Sigma^* \). The generation of a string is a process that has two steps:

- Initialisation: Choose, with respect to a probability distribution \( I \), one state \( q_0 \in Q \) as the initial state. Define \( q_0 \) as the current state.
• Generation: Let \( q \) be the current state. Decide whether to stop, with probability \( F(q) \), or to produce a transition \((q, a, q')\) with probability \( P(q, a, q') \), where \( a \in \Sigma \cup \{\lambda\} \) and \( q \in Q \) (\( \lambda \) is the empty string). Output \( a \) and set the current state to \( q' \).

One relevant question to be solved regarding PFSMs is how to calculate the probability assigned by a PFSM, \( A \), to a given string \( x \in \Sigma^* \). To deal with this problem, let \( \theta = (s_0, x'_1, s_1), (s_1, x'_2, s_2), \ldots, (s_{k-1}, x'_k, s_k) \) be a path for \( x \) in \( A \); i.e. a sequence of transitions so that \( x = x'_1 x'_2 \ldots x'_k \) (note that, in general, \( |x| \leq k \) because some symbols \( x'_j \) may be \( \lambda \)).

The probability assigned to the path \( \theta \) is given by the following expression:

\[
p_A(\theta) = I_A(s_0) \cdot \left( \prod_{j=1}^{k} P_A(s_{j-1}, x'_j, s_j) \right) \cdot F_A(s_k)
\]  

(6)

In general, a given string \( x \) can be generated by \( A \) through multiple valid paths. Let \( \Theta_A(x) \) be the set of all the valid paths for \( x \) in \( A \). The probability of generating \( x \) with \( A \) is given by:

\[
p_A(x) = \sum_{\theta \in \Theta_A(x)} p_A(\theta)
\]  

(7)

If \( \sum_x p_A(x) = 1 \), then \( A \) defines a probability distribution \( D \) in \( \Sigma^* \). This is guaranteed if \( A \) is consistent. A PFSM \( A \) is consistent if all its states appears in at least one valid path of \( \Theta_A \).

We will finish this brief introduction on PFSMs describing the concept of best path \( \hat{\theta} \) for a string \( x \) in a given PFSM \( A \). The best path is given by the following expression:

\[
\hat{\theta} = \arg \max_{\theta \in \Theta_A(x)} \{p_A(\theta)\}
\]  

(8)

PFSMs can be used as stochastic error-correction models. Stochastic error-correction models based on the well-known concept of edit distance \([26]\) are implemented by means of PFSMs in \([7]\). Specifically, these PFSMs are built by concatenating error-correction models based on PFSMs for individual symbols.

Figure 1 shows an example of PFSM that works as an stochastic error-correction model for a given symbol \( a \) contained in the alphabet \( \Sigma \). We will note such a PFSM as \( A_a \). As can be observed, the figure shows transitions for the different edit distance operations, namely, insertions, substitutions and deletions. The edge associated to the emission of \( \lambda \) (the empty string) is represented with a dashed line.

Figure 2 shows the result of concatenating three PFSMs at symbol-level, so as to obtain an stochastic error-correction model for a text string \( x \in \Sigma^* \), where \( x = x_1 x_2 x_3 \). The resulting PFSM can be minimised (see \([7]\)), giving the PFSM that is shown in Figure 3.

The problem of estimating the parameters of stochastic error-correction models based on the concept of edit distance has been studied in \([38]\), where the use of the EM algorithm is proposed. Due to the great simplicity of the parameters to be estimated, one alternative to the application of the EM algorithm consists in the use of the so-called ad-hoc stochastic error-correction models \([27]\). This technique was initially proposed for its use in the field of optical character recognition (OCR) and consists in reserving a probability mass for the substitution of one symbol by itself and distributing the rest of the probability mass among the different edit operations between strings.
We also propose the application of an ad-hoc stochastic error-correction model. One possible way of defining it consists in assigning a fixed probability mass to the substitution of one symbol by itself $p = 1 - \epsilon$, and uniformly distributing the remaining probability mass among the rest of possible transitions:

$$p' = \frac{\epsilon}{2|\Sigma|} \quad (9)$$

where $2|\Sigma|$ represents the number of transitions that have been defined for each state (with the exception of the substitution of one symbol by itself).

Alternatively, the model described above can be refined by assigning different probabilities depending on the type of edit operation that is applied (insertion, substitution or deletion). For this purpose, we introduce three factors, $f_i$, $f_s$ and $f_d$, that are used to assign weights to the probabilities assigned to insertions, substitutions and deletions, respectively. Given the value of $\epsilon$ and the values for the weighting factors, the auxiliary quantity $c$ is defined as follows:

$$c = \frac{\epsilon}{(f_i|\Sigma|) + (f_s(|\Sigma| - 1)) + f_d} \quad (10)$$

The probabilities for the insertion, substitution and deletion operations: $p_i$, $p_s$ and $p_d$, respectively, can be expressed in terms of the quantity $c$ and the weighting factors $f_i$, $f_s$ and $f_d$:

$$p_i = f_i \times c \quad (11)$$

$$p_s = f_s \times c \quad (12)$$

$$p_d = f_d \times c \quad (13)$$

Figure 3: Reduced version of $A_x$. 

Figure 2: Error-correction model for string $x = x_1x_2x_3$, $A_x$. The model has been obtained by concatenating $A_{x_1}$, $A_{x_2}$ and $A_{x_3}$.
Once we have defined the task in which the stochastic error-correction models will be applied, the values of the weighting factors $f_i$, $f_s$ and $f_d$ can be established by means of a development data set and the MERT algorithm.

We will end this section about stochastic error-correction models based on PFSMs discussing some issues regarding the problem of finding the best sequence of edit operations for a given string.

Given an error-correction model for the string $x \in \Sigma^*$, $A_x$, and a string with errors, $x'$, we will be interested in finding the best sequence of edit operations that are needed to change $x$ into $x'$. This problem is equivalent to the problem of finding the best path in $A_x$. For this purpose, the well-known Viterbi algorithm (see for example [43] for more details) can be used.

Additionally, the problem of finding the best sequence of edit operations has been studied in a more general setting, where a PFSM and a stochastic error-correction model are given [5]. In such setting, the PFSM accounts for the set of different strings belonging to a given language, while the error-correction model accounts for the typical variations that the strings tend to exhibit with respect to their standard form as represented by the PFSM. Under these circumstances, Amengual and Vidal [5] propose simple extensions of the Viterbi algorithm that efficiently solve the problem of finding the best sequence of edit operations.

### 3.2.2 Alternative IMT Formalisation

We propose an alternative formalisation of the IMT paradigm in which the user prefix and the target sentence constitute separated entities. This allows us to introduce stochastic error-correction models in the statistical formulation of the IMT process.

The starting point of our alternative IMT formalisation consists in solving the problem of finding the sentence $y_{I_1}$ in the target language that, at the same time, better explains the source sentence $x_{J_1}$ and the prefix given by the user $y_p$. This problem can be formally stated as follows:

$$\hat{y}_{I_1} = \arg \max_{I,y} \{ Pr(y_{I_1} | x_{J_1}, y_p) \}$$

(14)

Using the Bayes rule we can write:

$$Pr(y_{I_1} | x_{J_1}, y_p) = \frac{Pr(y_{I_1}) \cdot Pr(x_{J_1}, y_p \mid y_{I_1})}{Pr(x_{J_1}, y_p)}$$

Since the denominator here is independent of $y_{I_1}$, the sentence $\hat{y}_{I_1}$ can be found by maximising the expression $Pr(y_{I_1})Pr(x_{J_1}, y_p \mid y_{I_1})$. We arrive then at the following equation:

$$\hat{y}_{I_1} = \arg \max_{I,y} \{ Pr(y_{I_1}) \cdot Pr(x_{J_1} \mid y_{I_1}) \cdot Pr(y_p \mid y_{I_1}) \}$$

(15)

Now we make the following naive Bayes assumption: given the hypothesised target string $y_{I_1}$, the strings $x_{J_1}$ and $y_p$ are considered statistically independent. Thus, we obtain the following expression:

$$\hat{y}_{I_1} = \arg \max_{I,y} \{ Pr(y_{I_1}) \cdot Pr(x_{J_1} \mid y_{I_1}) \cdot Pr(y_p \mid y_{I_1}) \}$$

(16)

In the previous equation the following terms can be found:

- $Pr(y_{I_1})$: measures the well-formedness of $y_{I_1}$ as a sentence of the target language. This distribution can be approximated by means of a statistical language model.
• $Pr(x^I_1 \mid y^I_1)$: measures the appropriateness of the sentence $x^I_1$ as a possible translation of $y^I_1$. This distribution can be approximated by means of a statistical translation model.

• $Pr(y_p \mid y^I_1)$: measures the compatibility of $y^I_1$ with the user prefix $y_p$. This distribution can be approximated by stochastic error-correction models that are adequately modified for its use in IMT.

It should be noted that the result of the maximisation given by Equation (16), $\hat{y}^I_1$, may not contain the prefix $y_p$ given by the user, since every possible target sentence $y^I_1$ is compatible with the user prefix with a certain probability. Because of this, the problem defined by Equation (16) is not equivalent to the problem of finding the best suffix in IMT.

To solve this problem, an additional assumption over the stochastic error-correction models is imposed. Specifically, they must be able to determine an alignment between a part of the target sentence $y^I_1$ and the user prefix $y_p$. The set of unaligned words of $y^I_1$, $U_{y^I_1}$, in an appropriate order constitute the suffix required in IMT. To simplify things, we can also assume that we monotonically align a prefix of $y^I_1$ with $y_p$. This implies that the reordering problem is left to the language and translation models (i.e. the stochastic error-correction model is used to determine monotonic alignments between the user prefix and a prefix of the non-monotonic target translations of the source sentence). Under these circumstances, the suffix required in IMT is also a suffix of $y^I_1$.

The probability given by stochastic error-correction models can be expressed in terms of a hidden alignment variable, $p$, as follows:

$$Pr(y_p \mid y^I_1) = \sum_p Pr(y_p, p \mid y^I_1)$$ (17)

According to Equation (17) and following a maximum-approximation, we can modify the problem stated in Equation (16) so as to obtain not only the sentence $\hat{y}^I_1$, but also the alignment variable, $\hat{p}$, which maximises the probability:

$$\left(\hat{y}^I_1, \hat{p}\right) = \arg \max_{I, y^I_1, p} \{Pr(y^I_1) \cdot Pr(x^I_1 \mid y^I_1) \cdot Pr(y_p, p \mid y^I_1)\}$$ (18)

Figure 4 shows how the IMT suffix is determined in our alternative IMT formalisation. The IMT system receives the source sentence, $x^I_1$, and the user prefix, $y_p$, as input, and generates the best translation $\hat{y}^I_1$ along with an alignment between a prefix of $y^I_1$ and $y_p$. The suffix $y_s$ is obtained from the set $U_{y^I_1}$ of words of $y^I_1$ that are not aligned with $y_p$.

The stochastic error-correction models used in the proposed IMT formalisation can be defined in many ways. Here, error-correction models based on PFSMs will be used. It is worthy of note that the stochastic error-correction models can be seen as explicit models of the users of the IMT system.

Error-correction models based on PFSMs described in section 3.2.1 require some modifications for its use in IMT, since we want to model the probability distribution $Pr(y_p \mid y^I_1)$, where $y_p$ is a prefix instead of a complete sentence.

As a starting point, a stochastic error-correction model based on PFSMs, $A_{y^I_1}$, is defined (see Figure 3 for an example of this kind of stochastic error-correction models); where $A_{y^I_1}$ has been obtained as the concatenation of the error-correction models for each one of the words of $y^I_1$.

To allow this error-correction model to be used in the IMT framework, we only have to introduce one simple modification in $A_{y^I_1}$. Specifically, we assume that $F_A(q)$ is a non-null fixed
he reservado una habitación sencilla

I have made a reservation for

I have booked a single room

(a, single, room)

a single room

Figure 4: Example of how the IMT suffix is determined in our alternative IMT formalisation.

Figure 5: Error-correction model based on PFSMs for IMT given the sentence \( y^I_1: B_{y^I_1} \). The states of the PFSM are labelled with the words of the target sentence \( y^I_1 \).

quantity for each possible state \( q \) contained in \( Q_A \). We will note the resulting PFSM as \( B_{y^I_1} \). Figure 5 shows how the error-correction model is defined, the states of the PFSM are labelled with the words of the target sentence \( y^I_1 \).

Let \( \theta = (s_0, x'_1, s_1, (s_1, x'_2, s_2), ..., (s_{k-1}, x'_k, s_k) \) be a valid path for \( y_p \) in \( B_{y^I_1} \), where \( s_i \) are states contained in \( Q_{B_{y^I_1}} = \{ y_0, ..., y_I \} \) and \( x'_i \) are words of \( y_p \) or the empty string \( \lambda \). Each transition of the path is associated to an insertion, a substitution or a deletion operation. The path \( \theta \) determines a monotonic alignment between a prefix of the target sentence \( y^I_1 \) and the user prefix \( y_p \). The alignment decisions depend on the edit operation that is applied:

- **Insertions**: correspond to transitions of the form \((y_i, x', y_i)\), where \( x' \) is a word of \( y_p \). When these transitions are added to \( \theta \), the word \( x' \) of \( y_p \) is not aligned with any word of \( y^I_1 \).

- **Substitutions**: correspond to transitions of the form \((y_i, x', y_{i+1})\), where \( x' \) is a word of \( y_p \). These transitions align the word \( y_{i+1} \) of \( y^I_1 \) with the word \( x' \) of \( y_p \).

- **Deletions**: correspond to transitions of the form \((y_i, \lambda, y_{i+1})\). When these transitions are added to \( \theta \), the word \( y_{i+1} \) of \( y^I_1 \) is aligned with the empty string.

The final state \( s_k \) of the path \( \theta \) will be associated to the position \( i \) of the last word of \( y^I_1 \) which accounts for the user prefix \( y_p \) (this last word and the previous words will be aligned with words of \( y_p \) or with the empty string). Therefore, the suffix \( y_s \) required in IMT will be determined by \( y^I_{i+1} \).

Among all the valid paths for the string \( y_p \) in \( B_{y^I_1}, \Theta_{B_{y^I_1}}(y_p) \), we will be interested in that of the maximum probability \( \hat{\theta} \), where:

\[
\hat{\theta} = \arg \max_{\theta \in \Theta_{B_{y^I_1}}(y_p)} \{ p_{B_{y^I_1}}(\theta) \} 
\]
Hence, the best path $\hat{\theta}$ for $y_p$ in $B_{y^1_I}$ not only allows us to approximate the probability distribution $Pr(y_p \mid y^1_I)$, but also to determine the part of $y^1_I$ that constitutes the suffix $y_s$ required in IMT.

It is worth noticing that the error-correction model for IMT defined here works at word level, but it could have been defined to work at character level instead. A character-level error-correction model would allow us to assign higher probabilities to the substitution of one word by another similar word. This advantage would be obtained at the cost of a higher time complexity. Alternatively, the proposed ad-hoc word-level error-correction model can also be replaced by a more complex word-level model which defines specific substitution probabilities.

To instantiate our proposed alternative IMT formalism, the models that approximate the probability distributions that are present in Equation 18 have to be appropriately chosen. $Pr(y^1_I)$ and $Pr(x^1_J \mid y^1_I)$ can be approximated by a language model and a translation model, respectively. Regarding the probability distribution $Pr(y_p, p \mid y^1_I)$, it has to be approximated by an error-correction model that is able to determine an alignment between the target sentence $y^1_I$ and the user prefix $y_p$.

The stochastic error-correction models based on PFSMs for IMT described above can be used to determine the required alignment between $y^1_I$ and $y_p$. Specifically, this alignment is given by a path $\theta$ in the PFSM. The final state of $\theta$ determines the position $i$ of the last word of $y^1_I$ that accounts for the user prefix. Therefore, the suffix $y_s$ is given by $y^1_{i+1}$.

Given the previous considerations, a particular instantiation of the proposed alternative IMT formalism can be defined as:

$$\left(\hat{y}^1_I, \hat{\theta}\right) = \arg \max_{I, y^1_I, \theta} \{p(y^1_I) \cdot p(x^1_J \mid y^1_I) \cdot p_{B_{y^1_I}}(\theta)\}$$

(20)

where $\theta$ is contained in the set of all possible paths for the string $y_p$ in $B_{y^1_I}$, $\Theta_{B_{y^1_I}}(y_p)$.

It should be noted that the IMT formalism based on the Bayes rule given by the previous equation can be replaced by another one based on the log-linear approach. The resulting log-linear model would be composed of the standard components used in fully-automatic SMT, plus one more component corresponding to the log-probability given by the stochastic error-correction model. The set of standard SMT components is the same used in [33], and includes a total of seven feature functions, namely, a language model, a sentence length model, source and target phrase length models, a distortion model and inverse and direct phrase-based models.

The search procedure formalised by Equation (20) can be implemented as a process with two steps:

1. Generate a word graph for the source sentence $x^1_J$. The word graph is generated only once at the first interaction of the IMT process.
2. Apply the stochastic error-correction model over the target sentences contained in the word graph so as to obtain the pair $(\hat{y}^1_I, \hat{\theta})$ of maximum probability.

3.2.3 Experiments

This section describes the experiments that we carried out to test our proposed IMT technique based on stochastic error-correction models. The experiments were performed using the Xerox and the EU corpora. The Xerox corpus consists of translations of Xerox printer manuals

---

3In some experiments reported in this chapter we show the time cost of the proposed algorithms, all the experiments were executed on a PC with a 2.40 Ghz Intel Xeon processor with 1GB of memory.
involving three different pairs of languages: French-English, Spanish-English, and German-English. The EU corpora [18] was extracted from the Bulletin of the European Union, which exists in all official languages of the European Union and is publicly available on the Internet.

We evaluated our IMT system by means of the KSMR measure [8]. This is calculated as the number of keystrokes plus the number of mouse movements plus one more count per sentence (aimed at simulating the user action needed to accept the final translation), the sum of which is divided by the total number of reference characters. In addition to this, we also measured the time cost in seconds per each interaction. Such cost is important in an IMT scenario since the system should react to user interactions in real time.

In Table 1 the IMT results for the Xerox corpora (for the three language pairs and both translation directions) using our proposed IMT system based on stochastic error-corrections models are shown. A monotonic SMT system was used to generate the word graphs that are required during the IMT process. MERT for the development corpus was performed to adjust the weights of the log-linear model. The last column of Table 1 shows the average time in seconds per iteration needed to complete a new translation given a user validated prefix. These times allow the system to work on a real time scenario.

Table 1: KSMR results for the three Xerox corpora, using an IMT system based on stochastic error-correction models. Word graphs were generated by means of a monotonic SMT system. MERT was performed. The average time in seconds per interaction is also reported.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>KSMR</th>
<th>s/inter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spa–Eng</td>
<td>21.2</td>
<td>0.010</td>
</tr>
<tr>
<td>Eng–Spa</td>
<td>19.8</td>
<td>0.010</td>
</tr>
<tr>
<td>Fre–Eng</td>
<td>41.2</td>
<td>0.012</td>
</tr>
<tr>
<td>Eng–Fre</td>
<td>37.5</td>
<td>0.013</td>
</tr>
<tr>
<td>Ger–Eng</td>
<td>41.0</td>
<td>0.012</td>
</tr>
<tr>
<td>Eng–Ger</td>
<td>42.7</td>
<td>0.012</td>
</tr>
</tbody>
</table>

We also performed non-monotonic experiments. Table 2 shows the KSMR results when translating the Xerox test corpora from English to the other three languages. Word graphs were generated by means of a non-monotonic SMT system. The weights of the IMT system were tuned using the MERT algorithm. The average time costs per each interaction are also shown. As can be seen, slight improvements with respect to the monotonic IMT system can be obtained, at the cost of higher interaction times.

Finally, in Table 3 a comparison of the results obtained by our phrase-based IMT system based on stochastic error-correction models (PB-SECM) with state-of-the-art IMT systems is reported (95% confidence intervals are shown). The comparison includes the KSMR results obtained by IMT systems using different translation technologies, namely, alignment templates (AT), stochastic finite-state transducer (SFST), and phrase-based models (PB) (see [8] for more details). Additionally, we also show the results of an additional IMT system which is based on partial phrase-based alignments (PSPBA) [31]. As can be seen, our system is competitive with

Table 2: KSMR results for the three Xerox corpora, using an IMT system based on stochastic error-correction models. Word graphs were generated by means of a non-monotonic SMT system. MERT was performed. The average time in seconds per interaction is also reported.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>KSMR</th>
<th>s/inter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eng–Spa</td>
<td>19.3</td>
<td>0.048</td>
</tr>
<tr>
<td>Eng–Fre</td>
<td>36.9</td>
<td>0.100</td>
</tr>
<tr>
<td>Eng–Ger</td>
<td>42.2</td>
<td>0.084</td>
</tr>
</tbody>
</table>
the SFST and the AT systems but underperforms the results obtained by the PB and PSPBA IMT systems. It is worth mentioning that the AT and the SFST systems are also based on word graphs and error-correction techniques to generate the suffixes required in IMT, as well as our proposed IMT system; specifically, these systems obtain the translation of minimum edit distance to the given prefix. By contrast, the PB and PSPBA IMT systems generate a new translation of the source sentence at each interaction of the IMT process instead of generating a word graph at the beginning. This allows such IMT systems to obtain better results but generally with higher time costs per interaction. Specifically, we were able to compute comparable time costs per interaction for the PSPBA technique. Such time costs were of tenths of a second per interaction instead of hundredths of a second.

Table 3: KSMR results comparison of our IMT system based on stochastic error-correction models and four different state-of-the-art IMT systems (word graphs were generated by means of a monotonic SMT system). 95% confidence intervals are shown. The experiments were executed on the Xerox corpora. Best results are shown in bold.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>AT</th>
<th>PB</th>
<th>SFST</th>
<th>PSPBA</th>
<th>PB-SECM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spa–Eng</td>
<td>24.0±1.3</td>
<td>18.1±1.2</td>
<td>26.9±1.3</td>
<td>19.6±1.1</td>
<td>21.2±1.2</td>
</tr>
<tr>
<td>Eng–Spa</td>
<td>23.2±1.3</td>
<td>16.7±1.2</td>
<td>21.8±1.4</td>
<td>17.6±1.1</td>
<td>19.8±1.3</td>
</tr>
<tr>
<td>Fre–Eng</td>
<td>40.5±1.4</td>
<td>37.2±1.3</td>
<td>45.5±1.3</td>
<td>37.0±1.4</td>
<td>41.1±1.4</td>
</tr>
<tr>
<td>Eng–Fre</td>
<td>40.4±1.4</td>
<td>35.8±1.3</td>
<td>43.8±1.6</td>
<td>34.4±1.2</td>
<td>37.5±1.2</td>
</tr>
<tr>
<td>Ger–Eng</td>
<td>45.9±1.2</td>
<td>36.7±1.2</td>
<td>46.6±1.4</td>
<td>39.5±1.1</td>
<td>41.0±1.2</td>
</tr>
<tr>
<td>Eng–Ger</td>
<td>44.7±1.2</td>
<td>40.1±1.2</td>
<td>45.7±1.4</td>
<td>39.2±1.1</td>
<td>42.7±1.1</td>
</tr>
</tbody>
</table>

3.2.4 Experiments with the EU Corpus

We executed experiments on the EU corpora using our IMT system based on stochastic error-correction models. Table 4 shows the obtained KSMR results when translating from Spanish, French and German to the English language. The word graphs required in the IMT process were generated by means of a monotonic SMT system. The weights of the log-linear models were tuned via MERT. The table also shows the average time costs per each interaction of the interactive translation.

As in previous sections, we also present a comparison between the results obtained by our IMT system and those obtained by state-of-the-art IMT systems following different translation approaches, including the alignment templates (AT), the stochastic finite-state transducers (SFST) and the phrase-based (PB) approaches to IMT. Again, we have also included the results obtained by our proposed IMT system based on partial phrase-based alignments (PSPBA). Table 5 shows the obtained KSMR results (95% confidence intervals are shown). Again, the PB and PSPBA IMT systems obtained the best results, and our PB-SECM IMT system outperformed the results of the AT and the SFST IMT systems, which are also based on error-correction techniques to generate the suffixes required in IMT. We also computed comparable time costs for the best technique (PSPBA) and found that our proposed technique was one order of magnitude faster.

Table 4: KSMR results for the three EU corpora, using an IMT system based on stochastic error-correction models. Word graphs were generated by means of a monotonic SMT system. MERT was performed. The average time in seconds per iteration is also reported.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>KSMR</th>
<th>s/inter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spa–Eng</td>
<td>26.9</td>
<td>0.021</td>
</tr>
<tr>
<td>Fre–Eng</td>
<td>23.2</td>
<td>0.027</td>
</tr>
<tr>
<td>Ger–Eng</td>
<td>31.9</td>
<td>0.024</td>
</tr>
</tbody>
</table>
Table 5: KSMR results comparison of our IMT system based on stochastic error-correction models and four different state-of-the-art IMT systems (word graphs were generated by means of a monotonic SMT system). 95% confidence intervals are shown. The experiments were executed on the EU corpora. Best results are shown in bold.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>AT</th>
<th>PB</th>
<th>SFST</th>
<th>PSPBA</th>
<th>PB-SECM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spa–Eng</td>
<td>33.3±1.3</td>
<td>23.8±1.0</td>
<td>31.1±1.3</td>
<td>21.9±1.0</td>
<td>26.9±1.0</td>
</tr>
<tr>
<td>Fre–Eng</td>
<td>28.6±1.2</td>
<td>21.5±1.0</td>
<td>28.0±1.2</td>
<td>19.5±0.9</td>
<td>23.2±1.0</td>
</tr>
<tr>
<td>Ger–Eng</td>
<td>38.1±1.4</td>
<td>31.7±1.0</td>
<td>39.1±1.5</td>
<td>28.2±1.2</td>
<td>31.9±1.1</td>
</tr>
</tbody>
</table>

3.2.5 Conclusions

We have presented an IMT technique based on stochastic error-correction models. Such technique follows an alternative formalisation of the IMT framework in which the user prefix and the target sentence generated by the system constitute separated entities. This alternative formalisation introduces stochastic error-correction models in the statistical formulation of the IMT process. This contrasts with existing IMT systems in which error correction is applied but not formally justified. The proposed IMT technique generates the suffixes required in IMT by partially aligning a prefix of the target hypotheses with the user prefix. Once the partial alignment is determined, the suffix is given by the unaligned portion of the target sentence. It is worth pointing out that stochastic error-correction models can be seen as explicit models of the user of the IMT system.

We carried out experiments to test the performance of our proposed IMT technique with error-correction. Specifically, we compared the KSMR results obtained by our system with those obtained by other state-of-the-art IMT systems. Our system outperformed the results of those systems which are based on word graphs. On the other hand, our system obtained worse results than other systems that do not use word graphs, but in those cases where we were able to compute comparable times, our proposed system was approximately one order of magnitude faster.

In the rest of the period scheduled for this task, we intend to extend the stochastic error-correction models in several ways. A promising research direction involves including IBM word-alignment models into the error-correcting procedure, with the purpose of computing more robust and reliable alignments between hypotheses and user prefixes. In addition, the experiments performed as of yet are only preliminary, and are reported over corpora which are not the standard corpora present in CASMACAT. Hence, future work also involves obtaining results with such corpora.

4 Task 2: Multi-modality in interactive translation prediction

4.1 Introduction

Typically, the post-editing MT or interacting with ITP systems is carried out with a specific text editor using the keyboard and occasionally the mouse. This approach has been proved to be efficient by the translation industry to the point that [16] proposes post-editing guidelines for translation agencies. However, the user needs to be in front of a desktop computer which imposes some restrictions regarding where and how the work is to be done. Laptop computers can also be used, although arguably performance could be diminished because of the use of uncomfortable laptop keyboards and track pads.

An alternative way of interacting with the system is an interface where the user can use a touch screen or an electronic pen (e-pen) to perform post-editing or ITP tasks. Although e-pen
interaction may sound impractical for texts that need a large amount of post-editing, there is a number of circumstances where it can be more comfortable. First, it can be well suited for post-editing sentences with few errors, as it is the case of sentences with high fuzzy matches, or the revision of human post-edited sentences. Second, it would allow to perform such tasks while commuting, travelling or sitting comfortably on the couch in the living room.

Two main challenges emerge when tackling the use of e-pen in MT and ITP. The first one concerns on-line HTR systems committing errors. As the user may weary of using a faulty e-pen system, HTR robustness needs to be improved. Second, e-pen interaction can be extended beyond handwriting word recognition to allow the use of specific gestures to improve user efficiency.

4.2 Robust on-line HTR by leveraging MT models

In Attachment B we study the use of contextual information in MT post-editing and ITP to improve the robustness of the on-line HTR regarding isolated word recognition. First, we construct a list of the words in the vocabulary and exclude the word the user is trying to correct, since we already know that that word is erroneous. Then, we can take advantage of the supervised prefix, which is used as left context to compute the language model probability. Also, we can interpolate linearly that probability with the probability of the IBM1 and IBM2 direct and inverse translation models. The results show that a nice improvement of 2% absolute classification error can be gained from combining all the sources of information. From all the models, IBM2 models have shown to perform better.

4.3 Study of e-pen gestures for ITP

E-pen interaction can be much more than handwriting, for instance e-pen gestures can be of great value (see Attachment C). There is already a ‘de facto’ standard for gestures for proof reading from which we have extracted the most promising gestures: substitutions, deletions, insertions and, transpositions. Furthermore, we have added a shift gesture to move phrases to specific places in the text (i.e., the user circles the phrase and draws an arrow to the final destination). Then, we have studied two e-pen post-editing approaches. In the first one, we consider substitutions, deletions, insertions and, shifts. The number of these operations to obtain a reference can be computed with the translation error rate (TER) [40]. In the second approach, we assume that the user is working with an ITP system. In this case, we have also considered transpositions. Preliminary results show that gestures allow for a more efficient e-pen interaction. Thus, they are worth introducing in the prototype.

4.4 Conclusions

It has been shown that e-pen interaction can be tightly integrated in the MT and ITP process. Furthermore, it has potential to improve user efficiency. Thus, we will explore alternative ways of e-pen interaction to find new gestures that can improve user productivity. In addition, we foresee that e-pen interaction can also improve ergonomics in some particular scenarios. However, an empirical evaluation involving human translators would be needed to test this hypothesis. All in all, e-pen interaction seems quite promising, especially in a world full of touch screen devices.

In the rest of the time scheduled for this task, we intend to extend the on-line HTR system by allowing multi-word corrections, instead of just single-word, as is the case now. This would allow the final user much more flexibility when correcting a text, but leads to a much more complex modelling of the user’s interactions because interactions between different words might appear...
and will need to be modelled: in the single-word scenario, it is obvious which source word needs to be aligned with the word being written, but in the multi-word case it is not so evident any more. Concerning the e-pen gesture interface, the experiments reported so far were performed by simulating the interaction environment, with the purpose of assessing its functionality before implementing it within the CASMACAT workbench. Since the results obtained so far match the expectations, the e-pen gesture interface will be implemented into the CASMACAT workbench in the remainder of the time scheduled for this task, with the purpose of evaluating the e-pen technology in the third field trial. In addition, the experiments performed as of yet are only preliminary, and are reported over corpora which are not the standard corpora present in CASMACAT. Hence, future work also involves obtaining results with such corpora.

5 Task 4: Prediction from Parse Forest

5.1 Introduction

Although phrase-based translation models are still the most common approach within Statistical Machine Translation, many grammar-based models have been proposed lately. Such models use hierarchical and syntactic information in order to get good quality translations. Grammar-based translation models have been widely used, specially for pairs of languages with strong differences in the syntactic structure, such as English-Chinese.

Several algorithms have been proposed for Interactive Machine Translation using the models provided by the phrase-based approach. However, such algorithms do not apply to the hierarchical search graphs of the grammar-based models.

We report on two ongoing efforts to develop interactive machine translation for grammar-based models: (1) exact search for partial translations and (2) approximate search for user prefixes. The first effort is mainly carried by UPVLC and the second by UEDIN, and there are obvious synergies. Achieving a grammar-based ITP system would allow for very different interaction schemes, that move away from the conventional left-to-right user model. Nevertheless, evaluating such schemes goes beyond the scope of CASMACAT, and thus grammar-based models will not be evaluated in the field trials.

5.2 Exact Search for Partial Translation

As syntax-based decoder we used an Inversion Transduction Grammar (ITG) system described in detail in [14]. For this task we used the theoretical formalism of translation parse forests in order to represent the translation process carried on in a grammar-based decoder. We also define several algorithms over parse forests to obtain the translation of highest probability given a partial translation hypothesis. It must be noted that the partial translation hypothesis can represent any segment of the resulting sentence, not necessarily the prefix. Thus, we allow new interaction paradigms different to the traditional left-to-right interactive translation process.

5.2.1 Model

A translation parse forest is a Hypergraph (a generalization of graphs in which each edge can connect 2 or more vertexes) that represents in a compact way the n-best candidates for the translation of a sentence using a synchronous model. Every translation forest has a set of vertexes $\mathcal{V}$ and a set of edges $\mathcal{E}$.

Each vertex in a translation forest is characterised by a non-Terminal symbol (from the synchronous grammar), the start and the end positions of the source language sentence span covered
and the start and the end of the translated phrase \((n - 1 \text{ words})\) useful for the computation of the n-gram language model (language model context). From now on, we will represent the edges using capital letters (representing the non terminal) with a sub-index and a super-index (representing the start and the end of the source span) and the language model information between brackets. For example, \(A_4^1\)(did not...slap the) represents a vertex with the non-terminal symbol \(A\) that covers the input span from position 1 to position 4 the translated phrase starts with "did not" and ends with "slap the". For clarity, in some cases we represent the vertexes just as its non-terminal.

Each edge represents the use of one rule of the grammar and has attached one probability. For \(n\)-ary rules we will represent the rules as tuples of \(n + 1\) elements. The first element will be a vertex that results from the combination of application of the rule on the other \(n\) elements. For example, if we have the rule \(NP \rightarrow \text{la NN JJ — the JJ NN}\), there will be an edge \((NP_9^6(\text{the}...\text{witch}), \text{NN}_8^7(\text{witch}...\text{witch}), \text{JJ}_9^8(\text{green}...\text{green}))\).

The details about the translation process using Inversion Transduction Grammars, and thus obtaining the parse forests, can be found in ?? ITGs are binary synchronous grammars. Thus, each edge connects just three vertexes. We say that a vertex \(V_{ji}(a..b)\) is a leaf vertex, and we denote it by \(V_{ji}(a..b) \in L\), when there is at least one phrasal production in the ITG of the form \(V \rightarrow s_{ji} \mid a \beta b\) where \(s_{ji}\) is the source language span from \(i\) to \(j\), and \(\beta\) is a string of target language terminal symbols. We say that a vertex \(V_{ji}(a..b)\) is an Inner vertex, and we denote it by \(V_{ji}(a..b) \in I\) when it appears in at least one edge as the uppermost vertex. It must be noted that \(L \cap I \neq \emptyset\) due to the presence of phrasal productions. From now on, we will focus in the use of ITG translation forests for CAT.

### 5.2.2 Algorithms for Interactive Translation From Parse Forests

Translation Forests are compact representations of the n-best translation trees. We call \(\delta(V)\) the maximum probability of all the sub-trees that yield from the vertex \(V\). The computation of \(\delta\) for all the vertexes can be efficiently computed by means of dynamic programming (see Algorithm 1).

**Algorithm 1** Efficient computation of \(\Pr(\delta)\).

\[
\begin{align*}
\{ & \text{Initialisation} \} \\
\text{for all } V_{ji}^i(a..b) \in L \text{ do} & \quad \delta(V_{ji}^i(a..b)) = \max_{\gamma \in \Delta} \Pr(V \rightarrow s_{ji} \mid a \beta b) \\
\text{end for} \\
\{ & \text{Recursion} \} \\
\text{for } l = 2 \text{ to } |s| \text{ do} & \quad \text{for all } V_{i+l}^{i+l} \in V \text{ do} \\
& \quad \text{for all } (V_{i+l}^{i+l}, W, X) \in E \text{ do} \\
& \quad \quad \delta(V_{i+l}^{i+l}) = \max(\delta(V_{i+l}^{i+l}), \Pr(V_{i+l}^{i+l}, W, X)\delta(W)\delta(X)) \\
& \quad \text{end for} \\
& \quad \text{end for} \\
\text{end for}
\end{align*}
\]

It must be noted that if the edges chosen in the maximisation are stored properly, the tree with maximum probability can be easily retrieved. From such tree we can get the translation with maximum probability.

A possible interaction with the system is the correction by the human translator of the non-terminal and the corresponding correct translated phrase for a span of the input sentence. For that reason we are interested in the computation of the tree with maximum probability.
in the parse forest that assume those inputs as correct. That means that the translation tree must contain one vertex that fulfils such restrictions. We denote the probability of the tree with maximum probability that yields from a vertex as $\gamma_{t,s,e,n}(A)$, where $t$ is the translated phrase, $s$ and $e$ are the start and the end of the input span, and $n$ is the non-terminal symbol corresponding to such span.

Algorithm 2 computes efficiently $\gamma(t,s,e,n)$. Note that the $\delta$ can be precomputed for all the vertexes of the translation forest. For this algorithm we use a queue of vertexes and its methods `insert()` (inserts an element or a set of elements at the end of the queue) and `top()` (gets the first element and remove it from the queue). It must be noted that the set of a non-terminal $n$, a span of the source language sentence (with $s$ and $e$ as starting and ending positions) and a translated phrase $t$, corresponds at most, to one vertex in the Translation Forest. We will denote such vertex as $V(t,s,e,n)$. The case where there is no vertex that fulfils such restrictions will be tackled using error correcting techniques.

```plaintext
Algorithm 2 Efficient computation of $\gamma(t,s,e,n)$.

{Initialisation}
Queue=\emptyset
Queue.insert(V(t,s,e,n))
{Recursion}
while Queue≠ \emptyset do
  V=Queue.pop()
  for all Edge E of the form (A,V,B) or (A,B,V) do
    $\gamma_{t,s,e,n}(A) = \max(\gamma_{t,s,e,n}(A), \gamma_{t,s,e,n}(V)\delta(B)\Pr(E))$
    if A∉Queue then
      Queue.push(A)
    end if
  end for
end while
return $\max(\gamma_{t,s,e,n}(S_{0}^{\infty}))$
```

The interaction described for the Algorithm 2 does not represent a realistic scenario. It requires too much information from the human translator. A more reasonable interaction process demands from the human translator just a partial translation of the sentence (or to mark one part of the target language sentence that has been correctly translated). The system has to find the translated sentence with highest probability from all those that contain the partial translation.

Algorithm 2 can be extended to allow this kind of interaction. The extended algorithm has two steps: search and decoding. During the search step, the system looks for the set of all the vertexes $V^*$ of the parse forest that have exactly the same language model context. The decoding step is exactly the same as in Algorithm 2 but instead of inserting just one vertex at the beginning, the queue must begin with all the vertexes from $V^*$.

### 5.2.3 Conclusions

In order to perform an interactive translation process using grammatical models, it is necessary to use a formalism different from the word graphs. In this case we have chosen translation parse forests, that represent the process of hierarchic/syntactic translation in a compact way. In order to perform the completion of a partial hypothesis, several algorithms have been proposed. During the rest of the period scheduled for this task, the work will be focused mainly on the experimental study of the algorithms proposed. In addition, other aspects of the interactive translation process will be addressed, such as the study of error correcting techniques to allow the algorithms to start with partial hypothesis that are not in the parse forest.
5.3 Approximate Search for User Prefix

Predictions of the best continuation of a user prefix have been previously proposed for phrase-based models. This typically involves search for an approximate match of the prefix in the search graph produced by a decoder, since retranslation is considerably more costly. Note that in this context "approximate search" refers to the search for an approximate match (based on string edit distance and model cost), not for an approximate method such as beam search. Indeed, we are developing an exact method that leads to exact solutions, i.e., solutions that find the optimal approximate match.

In phrase-based models, each vertex in the graph has an optimal path leading into it (which has to be matched by the user prefix) and an optimal path leading to a full translation. Simply stated, the approximate search problem seeks to find the vertex that best matches the user prefix. The optimal path leaving the vertex is the sentence completion prediction.\footnote{Note that it is a bit more complicated since the edge leading into the vertex (corresponding to a phrase transition) may be only partially matched, so the completion of the edge is part of the prediction.}

Algorithms such as the one described by \cite{24} compute for each vertex a triplets of:

1. number of user prefix words consumed
2. edit cost
3. optimal model cost

If a vertex can be reached with a value for (1) we return the optimal value for (2) and (3).

In contrast, a vertex in the hyper graph generated by a grammar based model may be used in quite diverse ways. For instance, it may appear at the beginning of a sentence, or in the middle, or the end. Hence, we need to find optimal edit costs (edit distance and model cost) for any substring of the prefix. Finding a vertex that matches the last word of the user prefix is also not enough, we need to find the optimal derivation that includes this vertex and that matches the user prefix optimally.

5.3.1 Formal Definition

Let $W = w_1..w_n$ be the user input and $v \in \mathcal{V}$ a vertex.

Let $\Gamma(v)$ be the set of all derivations leading to the vertex $v$. Each derivation $\gamma \in \Gamma(v)$ yields a string $s(\gamma)$ with a model cost $m(\gamma)$.

For each string $s(\gamma)$ and user input $W$, we can compute the string edit distance $e(s(\gamma), W)$, typically the Levenshtein distance which counts insertions, deletions, and substitutions of words as 1 error, and word matches as 0 error.

Furthermore, for each prefix $s_k(\gamma)$ of length $k$ of $s(\gamma)$ and each substring $w_i..w_j$ (with $1 \leq i \leq j \leq n$), we can also compute the edit distance $e(s_k(\gamma), w_i..w_j)$.

We are concerned about both string edit distance $e$ and model cost $m$. We define the minimum for the pair of edit cost and model cost $(e, m)$ as

$$\min((e_1, m_1), (e_2, m_2)) = \begin{cases} (e_1, m_1) & \text{if } e_1 < e_2 \\ (e_2, m_2) & \text{if } e_1 > e_2 \\ (e_1, m_1) & \text{if } e_1 = e_2 \text{ and } m_1 \leq m_2 \\ (e_2, m_2) & \text{otherwise} \end{cases} \quad (21)$$
We define the function $B : (v, i, j) \rightarrow (e, m)$ as a mapping between a triplet of vertex $v$, start and end positions $1 \leq i \leq j \leq n$ to an optimal edit cost $e$ and model cost $s$ as

$$B_W(v, i, j) = \arg\min\gamma \{(e(s_k(\gamma), w_i..w_j), m(\gamma))\} \quad (22)$$

For all root vertices $v_r$ that span the entire input sentence, we compute the best matching derivation $B_W(v_r, 1, n)$, and chose the best derivation $\gamma_B$ among them. The sentence completion prediction follows the longest $s_k(\gamma_B)$ with optimal edit cost.

5.3.2 Example

See Figure 6 for a simple example forest generated by a synchronous context free grammar (SCFG). In this graph, each vertex has only one hyper edge, so we collapse these terms in the following description.

The best derivation $\gamma_B$ is $(2,4,6,5,7)$, yielding the string *the mouse chases the cat* with a model cost of 4.2. If the user enters any prefix of this string, this derivation will be returned to provide the best completion of the user input. However, if the user enters *the cat*, the best derivation $\gamma_B$ is $(1,3,4,6,7)$, yielding *the cat chases the mouse*.

Now consider the case that the user enters *the small cat*. This string cannot be found in any derivation. Still, we would like to see the sentence completion *chases the mouse*. How can we do this?

If we consider the vertices, we find that we can match *the small cat* with vertex $(7)$ (*the cat*) with string edit distance 1. This string is placed at the beginning of the translation of the full derivation $(2,4,6,5,7)$. Since it is a prefix of that full derivation, it is hence also a solution with string edit distance 1.

Consider the example input *the small cat chases*, we carry out the following processing steps, when limiting ourselves to an maximum edit cost of 1.
• Start with root vertex (1).

• The first symbol of (1) is a NP, so we recurse to vertex (7).
  (note: in real examples, there may be multiple possible edges to follow)

• Vertex (7) contains only words and we are still at the beginning of the user input, so we can compute the optimal ways to match any prefix of the user input.

<table>
<thead>
<tr>
<th>user start</th>
<th>user end</th>
<th>vertex end</th>
<th>edit cost</th>
<th>model cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>j</td>
<td>k</td>
<td>e(s_k(B_W((7), i,j)), w_i..w_j)</td>
<td>m(B_W((7), i, j))</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1.6</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1.6</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1.6</td>
</tr>
</tbody>
</table>

The three matches correspond to the cases where the first word (the) is matched and then (i) the vertex word cat is inserted, (ii) the vertex word cat is substituted with the word small and (iii) the user word small is deleted. Note that this table contains all matches that cover the entire vertex string (k=2), which allows processing of the next symbols up the tree.

• After completion of vertex (7) we return back to vertex (1).

• The next symbol in vertex (7) is the VP which can be only produced by vertex (3).

• Vertex (3) starts with a word, so we first explore the different ways to match it. We may have already consumed 1-3 words from the prefix, and we already incurred string edit distance cost of 1 in each case, so no further errors are allowed. This leaves only one remaining possibility:

<table>
<thead>
<tr>
<th>user start</th>
<th>user end</th>
<th>vertex end</th>
<th>edit cost</th>
<th>model cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>j</td>
<td>k</td>
<td>e(s_k(B_W((3), i,j)), w_i..w_j)</td>
<td>m(B_W((3), i, j))</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>2.4</td>
</tr>
</tbody>
</table>

• Since we reached the end of the user input (j = 4), we recurse back and consume no more symbols.

• Coming back to root vertex (1), we record that we found a solution with error 1 and model cost 4.7.

• We now have to do the same processing with root vertex (2), but will not find a better solution.

Please note some of the characteristics of this process: We match nodes only as needed by the top down search. We compute possible substring matches for each vertex as they are licenses by the maximum allowed edit cost. If we encounter the same vertex again with the same constraints (user input words already concerned, maximum edit cost allowed), we may re-use prior computations.

With a low allowed maximum edit cost, the process is very fast. So we will first try to find a match with maximum edit cost 0, then 1, then 2, etc., until a match is found.

5.3.3 Algorithm

The pseudo code for the algorithm is displayed in Algorithm 3. The main loop goes over all root vertices. The core of the code is the matching function match, which is called recursively to walk down the tree. Each call of the function operations of a vertex v with the assumptions that i user input words are already covered, and only a maximum edit cost $e_{\text{max}}$ is allowed.
Algorithm 3 Approximate string match for hierarchical models.

{Initialisation}
User prefix $W = w_1..w_n$
Set of vertices $V$, root vertices $V_R \subset V$
Edges $E$, $e \in E = (\text{lhs} \rightarrow \text{rhs}, v, v_{\alpha}..v_{\zeta}, m)$
Results $R = \epsilon$
{Main loop}
for $e_{\text{max}} = 0$ to $n$ do
   for all $v_r \in V_R$ do
      $R = R \cup \text{match}(v_r, 0, e_{\text{max}})$
   end for
   break if $R \neq \epsilon$
end for
return $\max(R)$
{Core Matching Function}
function $\text{match}(\text{vertex } v, \text{covered } i, \text{max edit cost } e_{\text{max}})$
Result set $R = \epsilon$
for all edges $e \in E$ with $v(e) = v$ do
   Intermediate result set $R' = \{(i, 0, m(e))\}$
   for all symbols $\sigma$ in $e$ do
      Next intermediate result set $R'' = \epsilon$
      for all intermediate results $r = (i', e_{\text{added}}, m) \in R'$ do
         if $\sigma$ non-terminal then
            Sub result set $R_S = \text{match}(v_\sigma, i', e_{\text{max}} - e_{\text{added}})$
            for all sub result $r_s \in R_S$ do
               $R'' = R'' \cup r + r_s$
            end for
         else
            add all allowable string edit extensions of $r$ to $R''$
         end if
      end for
      $R' = \{r \in R'' | i(r) < n\}$
      $R = R \cup \{r \in R'' | i(r) = n\}$
   end for
end for
$R = R \cup R'$
end for
return $R$
end function
The `match` function returns a set of results which reflect the function $B_W(v, i, j)$ we defined above. Within the function, partial or complete results are identified by the last input word currently covered $i'$, the edit cost added during the processing of this vertex $e_{\text{added}}$, and the model cost of applying the edge $m$.

There are three internal loops: (1) each edge $e$ below the vertex $v$ needs to be explored, (2) each symbol on the right hand side of the SCFG rule needs to be processed, (3) each intermediate result needs to be extended. Processing a non-terminal symbol triggers a recursive call to process the child vertex $v_\sigma$. Processing a word requires the exploration of possible edit steps (glossed over in the pseudo code), which are: a match, a substitution, an insertion (consumption of edge word), or any number of deletions — as long as the maximum allowed edit cost is not exceeded. Each edit operation leads to a new intermediate result.

A final result is separated out during processing (into $R$). Final results cover all user input words, so no further extensions are possible.

If calls to the `match` function in the main loop return results, then no search with higher edit cost is attempted, but the best result is returned.

The pseudo code omits some essential steps for brevity’s sake: Results obtained from calls to `match` are cached so that the function does not replicate them. Also, the path through the parse forest for each result must be tracked, so that the corresponding sentence completion can be produced.

5.3.4 Refinements

The worst case complexity of the algorithm is $O(n^4)$ where $n$ is the length of the input sentence, since the number of vertices is bound by $O(n^2)$, each vertex may be matched at each start word position (of $n$) and reach end word position (of $n$). Other factors such as the pop limit (number of vertices per span), the number of symbols per rule, etc., are bound by constants.

We are currently testing the algorithm and profiling its behavior. We will explore compacting the graph by removing vertices that produce equivalent strings with higher model cost than others. Equivalence may also consider that all words not matching any of the user words can be treated the same way.

6 Task 5: New SMT Models for ITP

6.1 Phrase-based hidden semi-Markov models

6.1.1 Introduction

The work performed in this direction has focused on obtaining state-of-the-art results in SMT with a hidden semi-Markov model. The advantages of this model is that it has a strong theoretical foundation, in contrast to heuristically estimated phrase-based models. Both Viterbi and Baum-Welch draining algorithms have been implemented. Both training algorithms obtain very similar results in practise, suggesting that most of the probability mass is gathered into one bilingual segmentation.

Even though the results obtained are competitive, and under some circumstances even better than those obtained with Moses, work in this direction was not continued until obtaining a fully-fledged ITP system because of the high computational cost involved when training the PBHISMM, which would potentially render the final ITP system unresponsive.
In this section, we review the phrase-based hidden semi-Markov model (PBHSMM) for monotone machine translation \[6\]. The conditional translation probability, \(p(x|y)\), is modelled assuming that the monotonic translation process has been carried out from left to right in segments of words or phrases. For this purpose, both sentences should be segmented in the same amount of phrases. In order to represent the segmentation process, we use two segmentation variables for both source (l), and target (m) sentences.

The target segmentation variable \(m\) stores each target segment length at the position at which the segment starts. Therefore, if the target segmentation variable \(m\) has a value greater than 0 at position \(i\), then a segment made up of \(m_i\) words starts at this position \(i\).

The source counterpart of the target segmentation variable is the source segmentation variable \(l\). The source segmentation random variable accounts for the length of each source segment at the position at which its corresponding target segment starts. If the source segmentation variable \(l\) has a value greater than 0 at position \(i\), then the length of the source segment corresponding to the target phrase that starts at position \(i\), is \(l_i\). Recall that the length of the target segment that starts at \(i\), is given by \(m_i\).

The PBHSMM is defined as a full exploration of all segmentations

\[
p(x|y) = \sum_m \sum_l p(x,l,m|y) ,
\]

where \(x\) is the input sentence, and \(y\) is the target sentence. \(m\) ranges among all the possible target segment values for \(y\), and \(l\) ranges only on those values that are in accordance with \(m\).

The complete model in Eq. (23) is decomposed as follows

\[
p(x,l,m|y) = p(m|y) p(l|m,y) p(x|l,m,y) .
\]

All the probabilities in Eq. (24) are decomposed left-to-right. We detail the decomposition of the target segmentation probability since the remaining probabilities are similarly decomposed.

The probability of the target segmentation variable is given by

\[
p(m|y) = \prod_i p(m_i|m_{i-1},y) .
\]

We assume that each partial probability in Eq. (25) does not depend neither on \(y\), nor on both lengths \((I\) and \(J\)), and, hence, the probability \(p(m_i|m_{i-1},y)\) is modelled as follows

\[
p(m_i|m_{i-1},y) := \begin{cases} p(m_i) & m_{i-1} + 1 = i, m_i > 0 \\ 1 & m_{i-1} + 1 \neq i, m_i = 0 \end{cases}
\]

Finally the segmentation probability can be expressed as follows

\[
p(m|y) := \prod_t p(m_t)
\]

Similarly to the target segmentation modelling, the source segmentation probability is simplified to

\[
p(l|m,y) := \prod_t p(l_t|m_t) ,
\]

where we have assumed that the \(t\)-th source segment length \(l_t\) only depends on the corresponding \(t\)-th target segment length \(m_t\).
Finally, the emission probability is also decomposed left-to-right
\[ p(x | l, m, y) := \prod_t p(x(t) | y(t)) \quad , \tag{29} \]
where \( x(t) \) stands for the \( t \)-th “emitted” source phrase, and \( y(t) \) stands for the \( t \)-th target phrase; and where we assume that the emission of the source phrase \( x(t) \) only depends on \( y(t) \).

Summarising, the proposed (complete) conditional translation model in Eq. (24) is defined as follows
\[ p(x, l, m | y) := \prod_t p(m_t) p(l_t | m_t) p(x(t) | y(t)) \quad , \tag{30} \]
where \( p(m), p(l | m), \) and \( p(u | v) \) constitute the model parameters for all \( m, l > 0, \) and for all source \( u \) and target \( v \) phrases.

### 6.1.3 Recurrences

As it is common in HSMM, some helpful recurrences are defined in order to efficiently train the model. The forward recurrence \( \alpha_{tl} \) is defined as the following prefix probability
\[ \alpha_{tl} = \alpha_{tl}(x, y) := p_\theta(x^l_1, l, t | y) \quad , \tag{31} \]
where the events \("l, t"\) imply that a source (or target) phrase ends at position \( l \) (or \( t \)) in the input (or output, respectively).

The prefix probability in Eq. (31) is recursively computed as follows
\[ \alpha_{tl} = \begin{cases} 1 & t, l = 0 \\ \sum_{t', l'} \alpha_{t'l'} p(t' - t) p(l' - l | t' - t) p(x^l_{t'+1} | y^l_{t'+1}) & 0 < t \leq I, 0 < l \leq J \\ 0 & \text{otherwise} \end{cases} \tag{32} \]
where, in the second case, \( t' \) ranges from 0 to \( t - 1 \), and \( l' \) ranges from 0 to \( l - 1 \).

The backward recurrence \( \beta_{tl} \) is defined as the following suffix probability
\[ \beta_{tl} = \beta_{tl}(x, y) = p_\theta(x^J_{l+1} | l, t, y) \quad , \tag{33} \]
where \( l, t \) implies that a source (or target) phrase ended at position \( l \) (or \( t \)) of the input (or output, respectively).

The suffix probability in Eq. (33) is recursively computed as follows
\[ \beta_{tl} = \begin{cases} 1 & t, l = I, J \\ \sum_{t', l'} \beta_{t'l'} p(t' - t) p(l' - l | t' - t) p(x^J_{l+1} | y^J_{l+1}) & 0 \leq t < I, 0 \leq l < J \\ 0 & \text{otherwise} \end{cases} \tag{34} \]
where, in the second case, \( t' \) ranges from \( t + 1 \) to \( I \), and \( l' \) ranges from \( l + 1 \) to \( J \).

Finally, the probability for a given bilingual pair \((x, y)\), can be computed by means of the forward or backward recurrences as follows
\[ p(x | y) = \alpha_{IJ} = \beta_{00} \quad . \tag{35} \]

The Viterbi recurrence answers the question of finding the best segmentation for a given bilingual pair \((x, y)\), i.e.,
\[ (\hat{l}, \hat{m}) = \arg \max_{l, m} \{ p(x, l, m | y) \} \quad . \tag{36} \]
The Viterbi recurrence is defined as follows
\[
\delta_{tl} = \max_{T, I^T, m^T_{l,T}} \{ p_{\theta}(x^T_{1,l}, I^T_l, m^T_l \mid y) \} ,
\]
where \( I^T_l, m^T_l \) are required to end at positions \( l \) and \( t \) respectively. This recurrence can be efficiently computed by the following recursive equation
\[
\delta_{tl} = \begin{cases} 
1 & t, l = 0 \\
\max_{t', t'} \delta_{t' t'} p(t' - t) p(t' - l \mid t' - t) p(x^T_{t+1} \mid x^T_{t+1}) & 0 < t \leq I \\
0 & 0 < l \leq J \\
& \text{otherwise}
\end{cases}
\]  
(38)

Note that a trace-back of the decisions made to compute \( \delta_{IJ} \) provides the solution to Eq. (36), i.e., the maximum probable bilingual segmentation.

All the proposed recurrences share the same asymptotic computational complexity. In order to compute these recurrences, a matrix of \( O(IJ) \) elements is needed. The computational complexity required to fill this matrix is \( O(I^2J^2) \). However, the complexity is reduced to \( O(IJP^2) \), if the phrases are constrained to a maximum source and target phrase length of \( P \).

### 6.1.4 Training

In this section, we give a description of the common training algorithms with respect to a collection of training translation pairs \( \{(x_N, y_N)\}_{n=1}^N \), that is to say: the Baum-Welch algorithm \( [32] \), and the Viterbi algorithm \( [35] \). Both algorithms are instantiations of the EM algorithm \( [12] \).

If an initial parameter set is given \( \theta^{(0)} \); then the EM iteratively refines this initial guess by alternatively applying two basic steps: the E(xpectation) step and the M(aximisation) step. These steps are applied until a maximum number of iterations are computed, or until the increment in the log-likelihood between two steps goes below a given threshold. Since the log-likelihood function induced by the PBHSMM is not convex, the EM training computes a local optimum. We must keep in mind that a wrong initial point \( \theta^{(0)} \) can ruin the system performance by making the EM to converge to a inappropriate local optimum.

In order to simplify notation, it is helpful to define the probability of using the source phrase \( x^T_{l+1} \) and the target phrase \( y^T_{l+1} \) when segmenting a given sample \( (x, y) \) as follows
\[
\gamma_{l t' t''} = \frac{\alpha_{lt} p(t' - t) p(t' - l \mid t' - t) p(x^T_{l+1} \mid x^T_{l+1}) \beta_{t' t''}}{p(x, y)}
\]
(39)

with \( l < l' \) and \( t < t' \), and where \( p(x, y) \) is computed as shown in Eq. (35).

**Baum-Welch training** In this case, the E step requires the computation, for each pair \( (x_n, y_n) \), of the sample versions of \( (31) \), and \( (33) \); as well as the fractional counts per sample. These sufficient statistics are computed using the parameters obtained from previous iteration, \( \theta^{(k)} \). The M step consists in estimating a new set of parameters \( \theta^{(k+1)} \), using the recurrences computed in the E-step.

The phrase dictionary is estimated as follows
\[
p^{(k+1)}(u \mid v) = \frac{N^{(k)}(u, v)}{\sum_{u'} N^{(k)}(u', v)}
\]
(40)

with
\[
N^{(k)}(u, v) = \sum_n \sum_{l, l'} \sum_{t, t'} \gamma^{(k)}_{n l t' t'} \delta(x_n(l, l'), u) \delta(y_n(t, t'), v)
\]
(41)
where we use the notation $z(l, l')$ to refer to $z_{l+1}^{l'}$, and where $\delta(a, b)$ stands for the Kronecker delta function which is to 1 if $a = b$ and 0 otherwise.

The target phrase length probabilities are estimated as follows

$$p^{(k+1)}(m) = \frac{N^{(k)}(m)}{\sum_{m'} N^{(k)}(m')} ,$$

(42)

with

$$N^{(k)}(m) = \sum_n \sum_{l,l'} \sum_t \gamma^{(k)}_{n,t,l,(t+m),l'} ,$$

(43)

where $m$ is a target phrase length.

Finally, the source phrase length probabilities are estimated as follows

$$p^{(k+1)}(l | m) = \frac{N^{(k)}(l, m)}{\sum_{l'} N^{(k)}(l', m)} ,$$

(44)

with

$$N^{(k)}(l, m) = \sum_n \sum_{l'} \sum_t \gamma^{(k)}_{n,t,l,(t+m),(l'+l)} ,$$

(45)

where $l$ and $m$ denote a source and target phrase length, respectively.

**Viterbi training** In this case, the E-step requires the computation, for each training pair $(x_n, y_n)$, of the maximum likely segmentation $(l^{(k)}_n, m^{(k)}_n)$ by means of the Viterbi recurrence in Eq. (38). Then, the M-step consists in applying standard Maximum likelihood estimation to the sample completed with the segmentation variables, i.e.,

The main difference between the Baum-Welch training and the Viterbi training is that the former uses the contribution of all the possible segmentations whereas the latter uses only the contribution of the maximum probable segmentation as defined in Eq. (36). It is a well-known issue that the Viterbi training yields worse results than that obtained by the Baum-Welch training [37]. The main advantage of the Viterbi training is that it only requires the computation of one recurrence.

6.1.5 Experiments

We have carried out two types of experiments. The first set of experiments were designed to analyse the properties of the proposed model when used in a classical phrase-based model made up of only the inverse translation probability and a language model. The second set of experiments were designed to analyse the properties of the PBHSMM when used as a feature inside a log-linear model. To evaluate the quality of the translations, two error measures were used: bilingual evaluation understudy (BLEU) [36], and translation edit rate (TER) [40].

For the first set of experiments, we tested our model in the Europarl-20 corpus. This corpus is made up of all the English-to-Spanish Europarl-v3 [23] sentences with length equal to or less than 20. We randomly selected 5,000 sentences to the test set and 2,000 to the development set. In the second experimental setup, we used the previous corpus; and the Europarl-v3 [23]. This corpus is the standard dataset used in [25]. Table 6 shows some basic statistics of both corpora the Europarl-20 and the Europarl-v3.
Table 6: Basic statistics of the corpora.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Language</th>
<th>Training</th>
<th>Development</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>En</td>
<td>Sp</td>
<td></td>
</tr>
<tr>
<td>Europarl-20</td>
<td>sentences</td>
<td>304 897</td>
<td>2 000</td>
<td>5 000</td>
</tr>
<tr>
<td></td>
<td>avg. length</td>
<td>12.7</td>
<td>12.8</td>
<td>12.6</td>
</tr>
<tr>
<td></td>
<td>running words</td>
<td>3.83M</td>
<td>25.1K</td>
<td>63.8K</td>
</tr>
<tr>
<td></td>
<td>voc. size</td>
<td>37.0K</td>
<td>3.9K</td>
<td>6.3K</td>
</tr>
<tr>
<td></td>
<td>ppl (5-gram)</td>
<td>–</td>
<td>62.2</td>
<td>63.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sp</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sentences</td>
<td>2.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>avg. length</td>
<td>21.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>running words</td>
<td>15.7M</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>voc. size</td>
<td>113.9K</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ppl (5-gram)</td>
<td>–</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Europarl-v3</td>
<td>sentences</td>
<td>730 740</td>
<td>2 000</td>
<td>2 000</td>
</tr>
<tr>
<td></td>
<td>avg. length</td>
<td>29.3</td>
<td>30.0</td>
<td>30.2</td>
</tr>
<tr>
<td></td>
<td>running words</td>
<td>58.7K</td>
<td>60.6K</td>
<td>60.3K</td>
</tr>
<tr>
<td></td>
<td>voc. size</td>
<td>6.5K</td>
<td>8.2K</td>
<td>8.3K</td>
</tr>
<tr>
<td></td>
<td>ppl (5-gram)</td>
<td>–</td>
<td>76.3</td>
<td>78.3</td>
</tr>
</tbody>
</table>

Classical phrase-based models. In the first set of experiments we compared two systems: the proposed PBHSMM for which we implemented a decoding algorithm based on standard algorithm techniques [19]; and the Moses system [20] but constraining the model to a classical SMT system made up of two components: a phrase-based inverse model and a 4-gram language model. The language model was smoothed with the modified Kneser-Ney smoothing, and was computed with the standard tool SRILM [41]. In this set of experiments, we constrained the Moses system in order to define a fair translation baseline. In the following subsection, a comparison between both systems inside a log-linear model is carried out.

Since the proposed training algorithms require an initial guess, we have computed the standard word-alignment pre-processing. First, we computed the IBM word alignment models with GIZA++ [30], for both translation directions. We computed the symmetrisation heuristic and extracted all the consistent phrases [31]. Afterwards, we computed our initial guess by counting the occurrences of each bilingual phrase and then normalising the counts. Instead of using the Moses system to perform this initialisation task, we have implemented our own version of this process.

Table 7 summarises the results obtained for both translation directions with the Europarl-20 corpus. Since the training algorithm depends on the maximum phrase length we limited it to 4 words. Surprisingly, Viterbi training obtains almost the same results as the Baum-Welch training; this is probably because most of the sentences accumulate all the probability mass in just one possible segmentation. Maybe that is why our algorithm is not able to obtain a large improvement with respect to the initialisation. Note that since the proposed system and the Moses system use different phrase-tables, these two numbers should not be compared. The Moses baseline is only given as a reference and not as a system to improve. Approximately 4 iterations suffice to avoid over-fitting, and to maximises the system performance. The results show a small improvement over the initialisation. Although the improvement is very small, it is similar to the improvement obtained when extending the maximum phrase length as shown in Table 8. For instance, it can be seen that extending the maximum phrase length from 4 to 5 words incurs in the same improvement as performing 4 Viterbi iterations in the model. In most of the cases, the Viterbi/Baum-Welch training improves the translation quality in terms of TER and/or BLEU.

Log-linear models. In this set of experiments, we compared 3 systems: Moses, log-PBHSMM, and log-PBHSMM+Moses. All the systems used the Moses decoder [20] to perform the decoding process. Thus, the differences among the three systems lay in the features used in the log-linear model. The first system is the standard log-linear system trained with Moses composed by the following standard features:

31
Table 7: Results for the Europarl-20 corpus with a maximum phrase length of 4 in a classical phrase-based translation system. “Moses” stands for a system trained with Moses, and with the following features: an inverse translation model \( p(x | y) \); and a 4-gram language model, \( p(y) \). The PBHSMM stands for the proposed system trained with both Viterbi and Baum-Welch algorithms.

<table>
<thead>
<tr>
<th>Max. phr. len.</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>It.</td>
<td>TER</td>
<td>BLEU</td>
<td>TER</td>
<td>BLEU</td>
</tr>
<tr>
<td>En → Sp</td>
<td>0</td>
<td>58.6</td>
<td>24.1</td>
<td>57.7</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>58.3</td>
<td>24.1</td>
<td>57.7</td>
</tr>
<tr>
<td>Sp → En</td>
<td>0</td>
<td>56.1</td>
<td>25.7</td>
<td>56.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>56.4</td>
<td>25.5</td>
<td>55.8</td>
</tr>
</tbody>
</table>

Table 8: Results in terms of TER and BLEU for the Europarl-20 corpus constrained by several maximum phrase lengths (Max. phr. len.). The table reports results at initialisation (0 It.) and after training 4 iterations (It.) of Viterbi training.
Table 9: Results for several translation systems on the Europarl-20 corpus.

<table>
<thead>
<tr>
<th>System</th>
<th>Distance-based reordering</th>
<th>Monotone (without reordering)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>En $\rightarrow$ Sp</td>
<td>Ter</td>
</tr>
<tr>
<td>Moses</td>
<td>56.7 ± 0.7</td>
<td>27.7 ± 0.7</td>
</tr>
<tr>
<td>log-PBHSMM</td>
<td>56.4 ± 0.7</td>
<td>28.1 ± 0.7</td>
</tr>
<tr>
<td>Moses+log-PBHSMM</td>
<td>56.4 ± 0.7</td>
<td>28.3 ± 0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ter</td>
</tr>
<tr>
<td>Moses</td>
<td>58.6 ± 0.7</td>
<td>26.1 ± 0.6</td>
</tr>
<tr>
<td>log-PBHSMM</td>
<td>57.6 ± 0.7</td>
<td>26.6 ± 0.6</td>
</tr>
<tr>
<td>Moses+log-PBHSMM</td>
<td>58.6 ± 0.7</td>
<td>26.4 ± 0.7</td>
</tr>
</tbody>
</table>

Table 9 shows the results in terms of Bleu and Ter for these systems using the Europarl-20 training corpus. Instead of computing a single figure, we computed the confidence interval at 95% [22].

Table 10 is the same as Table 9 but with the Europarl-v3 corpus. In this case, we constrained the maximum phrase length to the standard length of 7 words. This corpus has the peculiarity that the development and test sets are not distributed according to the training set probability distribution. This can be easily checked in Table 6 by comparing the average sentence lengths in each partition.

6.1.6 Conclusions

The results obtained in SMT with the PBHSMM show to be competitive, and even significantly better that those obtained with a comparable (although constrained) version of Moses, whenever the full PBHSMM system was used (i.e., training and PBHSMM-specific decoder). Moreover, adding the PBHSMM as an additional feature to the phrase-table when decoding with Moses proves to provide stable benefits, although not statistically significant.
6.2 Finite-state models

6.2.1 Introduction

In the previous work in TT2 project, finite-state models were employed as translation models for ITP. However those models were not built on the basis of a phrase-based approach, a well established SMT modelling technique nowadays. We have been working on adapting those finite-state models to a phrase-based approach following a log-linear framework similar to that in Moses. Nevertheless, a monotone translation framework has been adopted so far.

Even though finite-state models were not in the initial planning of the project, recent developments led to the idea that they could actually provide a complementary framework for providing the ITP engine with different SMT systems. For this reason, work on finite-state was performed during the first year of the project, with the purpose of determining whether such possibility was actually feasible.

6.2.2 Development

A finite-state approach to monotone Moses has been implemented, where seven models in log-linear combination are considered: a direct and an inverse phrase-based translation probability model, a direct and an inverse phrase-based lexical weighting model, phrase and word penalties, and a target language model.

Five out of these models are based on phrase-based scores which are organised under a phrase-table. These models can also be implemented by means of phrase-based weighted finite-state transducers on the basis of the Viterbi algorithm. The word penalty can also be equivalently redefined as another phrase-based score, similar to the five others, which allows us to
constitute a translation model $T$ composed of the union of six parallel phrase-based weighted finite-state transducers that are constrained to share the same monotonic bilingual segmentation.

A backoff $n$-gram model for the target language $\mathcal{L}$ can be represented as an identity weighted finite-state transducer where the probability of a given sentence is also modelled on the basis of the Viterbi algorithm. The whole log-linear approach to Moses is attained by means of the on-the-fly composition of both weighted finite-state transducers, $T \circ \mathcal{L}$. See [Attachment D] for further detail.

6.2.3 Conclusions

Even though the results achieved by our finite-state log-linear approach to phrase-based SMT are comparable to those obtained by monotone Moses, the comparison with a fully-fleged, state-of-the-art SMT system with reordering would be less convincing. For this reason, this work was finally discarded for its application within an ITP framework, considering work on this direction finalised within the CASMACAT project.
References


[16] TAUS in partnership with CNGL. Post-editing guidelines, 2011.


38


Attachment A

Task 2.1

Vicent Alabau, Alberto Sanchis, and Francisco Casacuberta.

On the optimal decision rule for sequential interactive structured prediction.

On the optimal decision rule for sequential interactive structured prediction

Vicent Alabau⇑, Alberto Sanchis, Francisco Casacuberta

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

A R T I C L E I N F O

Article history:
Received 7 December 2011
Available online 25 July 2012
Communicated by: G. Borgefors

Keywords:
Interactive pattern recognition
Minimum Bayes risk
Human interaction
Machine translation
Handwritten text recognition
Automatic speech recognition

A B S T R A C T

Interactive structured prediction (ISP) is an emerging framework for structured prediction (SP) where the user and the system collaborate to produce a high quality output. Typically, search algorithms applied to ISP problems have been based on the algorithms for fully-automatic SP systems. However, the decision rule applied should not be considered as optimal since the goal in ISP is to reduce human effort instead of output errors. In this work, we present some insight into the theory of the sequential ISP search problem. First, it is formulated as a decision theory problem from which a general analytical formulation of the optimal decision rule is derived. Then, it is compared with the standard formulation to establish under what conditions the standard algorithm should perform similarly to the optimal decision rule. Finally, a general and practical implementation is given and evaluated against three classical ISP problems: interactive machine translation, interactive handwritten text recognition, and interactive speech recognition.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Structured prediction (Parker et al., 2009) is a classification problem in which the output consists of structured labels (as opposed to independent labels), i.e., the output labels have dependencies on each other. Examples of structured outputs are natural language, DNA sequences, or an XML describing the layout analysis of a web page. Traditionally, SP has been approached as a fully automated procedure in which an input (\(x\)) is presented to a SP system and the SP system produces an output (\(y\)). As a post-processing, it is advisable for a human expert to revise the system output to amend the possible errors in order to produce the final output (or reference), \(r\). This post-editing (PE) process (depicted in Fig. 1a) is completely manual and all the supervision effort is delegated to the user.

The interactive structured prediction (ISP) framework (also known as interactive pattern recognition (Toselli et al., 2011)) was introduced in (Vidal et al., 2007) to reduce the cost of correcting the automatically generated output. In ISP, the user is introduced in the core of a SP system so that the system and the user can interact with each other to minimize the effort required to produce a satisfactory output (Fig. 1b represents the ISP interaction scheme). In ISP, an input \(x\) is given to the system, which outputs a possible hypothesis \(y\). Then, the user analyzes \(y\) and provides feedback \(f\) regarding some of the errors committed. Now, the system can benefit from the feedback to propose a new improved hypothesis. This process is repeated until the user finds a satisfactory solution, \(r\), and the process ends.

Traditionally, SP systems are designed following a decision rule to minimize output errors. However, this is not optimal for ISP since the decision rule should be formalized in terms of minimizing user interactions. Indeed, this fact was proved in (Oncina, 2009), where an alternative strategy is applied to a specific case of ISP (i.e., text prediction). Inspired by that work, here we provide an optimal decision rule for ISP that covers a broader range of common ISP problems in which the output depends on a structured input \(x\). This strategy is analyzed from a theoretical perspective and a practical decoding algorithm is developed to be used straightforwardly in many ISP tasks. In addition, we propose theoretically and empirically that the algorithm that has been used for ISP until now is a good approximation to the optimum for ISP.

The remainder of this article is organized as follows. First, the ISP framework is formalized in Section 2. Section 3 describes an optimal decision rule to minimize the number of interactions under the decision theory perspective. A general and practical algorithm is developed in Section 4. Finally, empirical results are shown in Section 5 and conclusions and future work are described in Section 6.

2. Sequential interactive structured prediction

Sequential interactive structured prediction (SISP) is a specific case of ISP where the user validates and/or corrects the system output in sequential order (typically left-to-right). This is espe-
cially interesting for many natural language processing (NLP) tasks, since humans usually listen, read, write, and talk in sequential order.

Let \( y^{(0)} = p^{(0)} \cdot s^{(0)} \) be a hypothesis at iteration \( i \) that is a concatenation of a correct prefix \( p^{(0)} \) of the solution \( r \) and a suffix hypothesis \( s^{(0)} \). Then, the protocol that rules SISP to obtain \( r \) can be formulated in the following steps:

1. Initially (\( i = 0 \)), the correct prefix is the empty string, \( p^{(0)} = \lambda \), and the system proposes a complete hypothesis \( s^{(0)} \).
2. At iteration \( i > 1 \), the user finds the longest prefix \( a^{(i)} \) of \( s^{(i-1)} \) that is error-free and corrects the first error in the suffix, which, let us assume, is at position \( k \), with \( r_k \).
3. A new extended prefix \( p^{(i)} \) is produced as a concatenation \( p^{(i-1)} \cdot a^{(i)} \) and the new introduced word \( r_k \).
4. Then, the system proposes a suffix hypothesis \( s^{(i)} \) that follows the prefix \( p^{(i)} \) established in the previous step.
5. Steps 1, 2 and 3 are iterated until, at some iteration \( i = I \) with \( I \leq |r| \), a correct solution is obtained, \( \hat{y}^{(I)} = r \).

Fig. 2 shows an example of this protocol for SISP. Note that, when the user introduces \( r_4 = \text{‘can not’} \), the system amends automatically the word ‘cannot’. Similarly, introducing \( r_7 = \text{‘in’} \) corrects ‘web’. Thus, in a PE system, the user would have needed to make five corrections, whereas just two corrections were needed in SIPS.

3. Optimal decision rule for SISP

Following the minimum classification error (MCE) (Duda et al., 2001), the optimum decision rule is the one that minimizes the average probability of loss (conditional risk) over a probability distribution \( \Pr(y) \). In classification problems, the human effort can be approximated by a cost 0 if the output is correct and a cost 1 if the user has to amend an erroneously classified sample by assigning the correct label. This function is known as the zero-one loss function and leads to the maximum-a-posteriori (MAP) decision rule (Duda et al., 2001): Let \( \hat{y} = \text{argmax}_y \Pr(y|x) \).

Typically, SISP systems have used an extension of MAP which has been successfully applied to several NLP tasks (Toselli et al., 2011). Thus, at steps 0 and 3 of the SIPS protocol, a suffix hypothesis \( s^{(0)} \) that continues a validated prefix \( p^{(0)} \) is generated using the following equation:

\[
\hat{s}^{(0)} = \text{argmax}_s \Pr(s|x, p^{(0)})
\]

where the output in the interaction (i) is \( \hat{y}^{(0)} = p^{(0)} \cdot \hat{s}^{(0)} \).

Nevertheless, the MAP rule seeks solutions with zero errors, whereas we would prefer a hypothesis that minimizes the number of human corrections.

3.1. The cost of interactively correcting the output

Traditionally, the cost of post-editing a sentence is measured in terms of the edit distance (i.e., the minimum number of substitutions, insertions, and deletions needed to transform the system output into the reference). A specific case is the Hamming distance, where only substitutions are needed since there is a one-to-one correspondence between the input and the output. Of course, the edit distance is a simplistic approach to assess PE effort. On the one hand, it is optimistic regarding the number of operations to make, since a human would hardly ever perform the operations with the minimum number of operations, especially in complex problems. On the other hand, the edit distance assigns the same cost to all the operations, regardless of the complexity of the problem and the cognitive effort needed to perform them. On the positive
side, the edit distance provides an automatic and intuitive measure. Consequently, it has been widely adopted for many NLP tasks as the PE cost.

Analogously, in ISP systems, the cost of interactively correcting a system output can be computed as the number of corrections (substitutions) needed to obtain the reference. Note that this is not equivalent to the Hamming distance since the part on the right of the output may change after each user correction. In this case, the cognitive effort is also neglected and the substitution cost is the same for all corrections. Besides, system suggestions may influence human corrections, since a good proposal could change a user’s opinion regarding what the correct solution is. In this sense, using a unique reference can be deemed as a pessimistic approach in problems with multiple correct solutions. Notwithstanding, this criteria can be considered to be a reasonably valid approximation.

In the following, we will denote with $y_k$ the $k$-th element in $y$ and $y_{j:k-1}$ as the substring from $j$ to $k - 1$ of $y$.

**Proposition 1.** Let $Pr(x | p^{(i)})$ be a posterior probability over the suffixes that continue $p^{(i)} = (y_1, \ldots, y_i)$. A suffix $s^{(i)} = (y_{i+1}, \ldots, y_{j})$ with the last symbol at position $i$, which minimizes the conditional risk of the number of interactions, can be obtained following this decision rule:

$$
y_k^{(i)} = \arg\max_{h} \sum_{r} Pr(y_k | s^{(i)} | x \cdot p^{(i)} \cdot y_{j:k-1})$

for $j = \ldots, I \land y_{j:k-1} \equiv S$  \hspace{1cm} (3)

where $s^{(i)}$ is a possible suffix for $p^{(i)}$, $y_{j:k-1}$, $y_k$ and $S$ is a special symbol that means the end of the hypothesis.

The algorithm works by constructing the output incrementally by appending individual labels. The decision of appending a new label is conditioned to previous labels, but it is independent of future decisions. The idea behind this is that, if the user amends a label at position $i$, $y_i$, all of the following labels in $p^{(i)}$ (i.e., $y_{i+1}, \ldots$) will be discarded in favor of a new suffix, $s^{(i)}$.

The proof of Proposition 1 provided in Appendix A.1 is inspired in previous work by Oncina (2009), who reached a similar algorithm. However, there are two major differences. First, in the SISP protocol proposed in (Oncina, 2009), the system predicts one symbol instead of complete suffixes. Second, Oncina (2009) presented the algorithm for text prediction (cf. autocompletion) where there is no dependency on a structured input $x$. Hence, if the dependence on the $x$ is dropped in Eq. (3), then it is possible to integrate the summation for all suffixes $s^{(i)}$, producing the optimum algorithm in Oncina (2009).

$$
y_k^{(i)} = \arg\max_{h} Pr(y_k | p^{(i)} | y_{j:k-1})$

for $j = \ldots, I \land y_{j:k-1} \equiv S$  \hspace{1cm} (4)

Thus, the resulting algorithm is greedy in the sense that it makes the decision locally, which is in contrast to Eq. (3) where the decision is taken globally. That is, the probability in Eq. (4) models only the next label, whereas in Eq. (3) it models whole suffixes. Unfortunately, for most SP problems, directly modeling the “local” posterior probability in Eq. (4) is still an unresolved problem, especially those with latent variables. Therefore, it is necessary to rely on the global models used in Eq. (3) and perform the sum explicitly over a (potentially) exponential number of suffixes.

### 3.2. Relation with the MAP decision rule

It has been mentioned that the MAP decision rule (Eq. (2)) has been extensively used for SISP. Although Eq. (2) is known to minimize the zero-one loss function for hypothesis suffixes, it would be interesting to analyze how it behaves with respect to Eq. (3).

**Proposition 2.** The MAP decision rule is equivalent to a maximum approximation to the optimal decision rule for SISP.

$$
y_k^{(i)} = \arg\max_{h} Pr(y_k | s^{(i)} | x \cdot p^{(i)} \cdot y_{j:k-1})$

for $j = \ldots, I \land y_{j:k-1} \equiv S$  \hspace{1cm} (5)

The proof of Proposition 2 can be found in Appendix A.2. It follows from a Bayes decomposition of Eq. (2) for each $y_k$. Proposition 2 has two main implications. Firstly, it provides a formalism for the traditional MAP approach as it can be seen as a maximum approximation to the optimum. That is especially convenient for models where Eq. (3) cannot be computed efficiently (exponential number of suffixes). Secondly, on non-smooth probability distributions where the mass probability is concentrated around the maximum, MAP performs almost as accurately as the optimum algorithm.

### 4. Decoding algorithm

The algorithm described in Proposition 1 has the difficulty that the sum over all possible suffixes must be done explicitly. In practice, this may be a major problem, since SP outputs are combinatorial by nature. Hence, to list all possible suffixes can be a hard problem. To deal with this problem, we propose a general practical algorithm for optimal decoding. The whole set of hypotheses (search space) will be represented by a word graph.

A word graph $G(x) = (\Sigma, Q, A, q_0, f)$ is a probability distribution over the hypothesis space given an input variable $x$. $G(x)$ is represented as a directed, acyclic, weighted graph where $\Sigma$ is a set of symbols and $Q$ is a set of nodes with $q_i$ as the initial node and $q_f$ as the final node. $A : \Sigma \times Q \times Q$ is a set of edges, $e = (y, u, v)$, which hypothesizes a label $y$ from $\Sigma$ from a start node $u$ to an end node $v \in Q$ with a score of $f(e, x)$. A path $w = (e_1, \ldots, e_n)$ is a sequence of connected edges that represents a complete hypothesis.

Given the input $x$, the posterior probability for a specific edge $e$ can be computed by summing up the posterior probabilities of all hypotheses of the word graph containing $e$. These posterior probabilities (here we use small $p$ to denote models instead of true probabilities) can be efficiently computed based on the well-known forward–backward algorithm (Wessel et al., 2001).

$$p(e | x) = p(y(u, v, u) | x) = \frac{\Psi(u)(e, x)\Psi(v)}{\Psi(q_f)}$$  \hspace{1cm} (6)

where the forward score $\Psi(u)$ for node $u$ is the sum of all possible paths from the initial node $q_i$ to $u$. Similarly, the backward score $\Psi(v)$ for node $v$ is the sum of all possible paths from $v$ to the final node $q_f$. Fig. 3 shows a (pruned) word graph obtained as the result of the translation of the input sentence for the example in Fig. 2, after the word posterior probabilities have been computed.

Now, we can conveniently introduce the prefix $p^{(i)}$ dependency in Eq. (6):

$$p(e | x, p^{(i)}) = p(y(u, v, u) | x, p^{(i)}) = \frac{\Psi_w(u)(f(e, x)\Psi(v)}{Z_{p^{(i)}}}$$  \hspace{1cm} (7)

Note that the prefix dependency affects the forward score $\Phi_w$. In this case, Eq. (7) is restricted to the sum of all paths from the initial node $q_i$ to $u$ for which the sequence of labels matches the prefix $p^{(i)}$. Also, the normalization factor $Z_{p^{(i)}}$ now only takes into account the mass probability of all the paths that have $p^{(i)}$ as a prefix.

Then, Eq. (3) can be easily computed by marginalizing over all of the edges with the word $y$ that follow $p^{(i)}$:

$$\sum_r Pr(y | s | x, p^{(i)}) \approx p(y | x, p^{(i)}) = \sum_u \sum_r p(y(u, v, u) | x, p^{(i)})$$  \hspace{1cm} (8)

Fig. 4 exemplifies how the state of the optimum algorithm changes when predicting the word at position $j = 4$ in iteration $(i = 0)$ for the example in Fig. 2. Note that an error is committed.
by the MAP approach Fig. 4a since it relies on the edge with the highest probability. In contrast, the optimum algorithm selects the set of edges with the same word whose sum is maximum, allowing the correct solution to be chosen.

5. Experimentation

First, we designed a simulated scenario. This allowed us to assess under what conditions the optimal decision rule outperformed the classical approach. For this purpose, we obtained the references from the Wall Street Journal (WSJ) database (Pallett et al., 1994). Then, for each reference we built a graph with 1000 hypotheses generated by introducing uniformly distributed random errors on the reference, with an error rate. Next, we assigned a score for each hypothesis assuming that the posterior probability followed an exponential distribution, \( k e^{-C_0k^n} \), where \( n \) is the number of the hypothesis being generated and \( kP_0 \) controls the peakedness of the distribution (the bigger the peakier).

In addition, three real world NLP tasks were selected: machine translation, handwritten text recognition, and automatic speech recognition. The problem of these three tasks consists of producing a translation or transcription \( y \) given an input signal \( x \). Formally, the problem can be solved by obtaining the sentence \( y \) that maximizes the posterior probability:

\[
y = \arg\max_y P(y \mid x) = \arg\max_y P(x \mid y) P(y)
\]

where \( P(y) \) is typically approximated by a language model, usually \( n \)-grams (Jelinek, 1997). In contrast, modeling \( P(x \mid y) \) is task dependent.

5.1. Machine translation

In machine translation (MT), \( x \) is a sentence in the source language. \( P(y \mid x) \) is usually approximated by phrase-based log-linear models (Och and Ney, 2002). The experiments were conducted on the Xerox corpus (Esteban et al., 2004) (English and Spanish documents), which is a collection of technical manuals. This corpus features approximately 700 k running words for training and 90 k running words for test. Test perplexities are 48 for English and 33 for Spanish. The size of the vocabulary in training is 8 k and 11 k, respectively, while the number of out-of-vocabulary words (OOV) is around 0.6%.

5.2. Handwritten text recognition

In handwritten text recognition (HTR), \( x \) is a feature vector that represents a handwritten sentence. In this case, however, \( P(x \mid y) \) represents morphological-lexical knowledge and is approximated by hidden Markov models (HMM) (Jelinek, 1997). HTR experiments were conducted on the “Cristo-Salvador” corpus (Romero et al., 2007), which was kindly provided by the Biblioteca Valenciana Digital (BIVALDI)². In the page version, the test set is composed of 491 samples corresponding to the last ten lines of each document page (4.5 k running words), whereas the training set is composed of the 681 remaining samples (6.4 k running words). The experiments were run on a closed vocabulary setting containing 3.4 k words.

5.3. Automatic speech recognition

The automatic speech recognition (ASR) problem is modeled in a very similar fashion to the HTR problem. In this case, however, \( x \)
represents a speech signal. Therefore, \( P(r | y) \) is a HMM model for the phonological-lexical knowledge. The experiments were performed using the WSJ database mentioned in the simulated experiments. Up to 81 h of training material (WSJ0 + WSJ1 partitions) were used to train speaker-independent HMMs. The test was composed of 213 sentences and 3.4 k running words with a perplexity of 168. The recognition was performed with the open vocabulary setup (64 k words) containing 1.6% OOVs.

5.4. Results

These corpora were evaluated based on two different measures. First, the PE effort was measured with the word error rate (WER). WER can be computed as the edit distance between the hypothesis and the reference normalized by the number of words in the reference. In contrast, SISP systems were evaluated with the word stroke ratio (WSR), which is the number of interactions normalized by the number of words in the reference. Note that both the edit distance and the number of interactions could have been computed at the character level as well. In that case, different costs would be attributed to correct shorter or longer predictions, but the influences of the word autocompletion feature that most editors have should also be taken into account. Besides, it may be argued that number of characters corrected seems to be more related to typing effort, whereas the cognitive effort would be more correlated to word units. Hence, word edit distance seems a more intuitive measure. We will denote the traditional MAP approach to SISP as SISP-MAP and the proposed approach as SISP-OPT.

Fig. 5 shows the evolution of the SISP-OPT decision rule as \( \lambda \) approaches one. It can be seen that, in this ideal scenario, when the distribution is smooth, the sum of different suffix hypotheses averages to obtain a much better result. However, as \( \lambda \) reaches one the distribution becomes so peaky that SISP-OPT is equal to SISP-MAP from that point on. The results for real tasks are presented in Table 1. It can be seen that the two SISP systems outperform their PE counterparts. With regard to the comparison between both SISP approaches, SISP-OPT always performs equal to or better than SISP-MAP. As explained in Section 3.2, SISP-MAP should be close or match SISP-OPT performance for non-smooth probability distributions, i.e., the probability mass is concentrated around the maximum. HMMs for HTR are structured input. Our approach extends previous work by allowing full suffix prediction instead of single symbols. This can be considered to be a relevant contribution since they represent the most frequent problems in SP. In addition, the maximum-a-posteriori approach was presented as a maximum approximation to the optimal decision rule.

The results for the real tasks have shown minimal improvements as anticipated in Proposition 2, but they are also supported by Schlueter et al. (2012). That paper study the relation between the MAP rule and the optimum rule for any integer-valued metric cost function for PE systems. They conclude that the use of optimal decision rules for task-related cost functions has a limited impact. Therefore, although the cost function for SIPS is not a metric by itself, the results indicate that those conclusions could be extrapolated to our case studies.

6. Conclusions

In this paper, we present an optimal decision rule for sequential interactive structured prediction (SISP) that generalizes the work on text prediction by Oncina (2009) to SISP problems that depend on a structured input. Our approach extends previous work by allowing full suffix prediction instead of single symbols. This can be considered to be a relevant contribution since they represent the most frequent problems in SP. In addition, the maximum-a-posteriori approach was presented as a maximum approximation to the optimal decision rule. Furthermore, a practical and general decoding algorithm was developed over word graphs. Experiments on different NLP tasks have shown that the MAP decision rule performs very similarly to the optimal one for non-smooth probability distributions, as it was expected. However, the optimum strategy has still been able to obtain minor improvements.

Further work should delve into the analysis of the optimal decision rule behavior. Directly implementing the optimal decision rule instead of using word graphs would probably lead to better improvements, since the sum is made over a wider range of hypotheses. It would also be interesting to test the results on other NLP tasks. Further research should especially concentrate on finding real tasks with smooth probability distributions so that the behavior of the optimal decision rule under more favorable conditions could be analyzed. Finally, the theoretical properties of the algorithm should also be studied further. Hopefully, that would allow determining under what conditions it is worthwhile to use the optimal decision rule. Thus, if improvements are not expected, then the use of non-optimal algorithms would be completely justified, given that, in practice, MAP algorithms are easier to compute.
Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under Grant agreement no. 287576 (CasMaCat), and from the Spanish MEC/MICINN under the MIPREV “Consolider Ingenio 2010” program (CDSD2007-00018) and iTrans2 (TIN2009-14511) project. It is also supported by the Generalitat Valenciana under grant ALMPR (Prometeo/2009/01) and GV/2010/067. The authors thank the anonymous reviewers for their criticisms and suggestions.

Appendix A. Proofs

A.1. Proof of Proposition 1

Proof.

Definition 1.

The SISP protocol has an associated cost function $C(y^0, j, r)$ that computes the cost of sequentially producing a reference $r$ from the label at position $j$ of the hypothesis $y^0$ at iteration ($i$). Using the abbreviated notation, $C^0_i$, the cost can be described as:

$$C^0_i = \delta_i (1 + C_i^{i+1}) + \delta_i' C_{i+1}^i \tag{A.1}$$

where $\delta_i' = \delta(y_i^0, r_i)$ is a Kronecker delta function that is 1 if $y_i^0 \equiv r_i$ and 0 otherwise, whereas $\delta_i$ is the negation of $\delta_i'$.

Following the MCE approach, an optimum algorithm for SIPS is one that minimizes the conditional expected value ($E(\cdot; \cdot)$) of $C^0_i$:

$$\langle y^0_1, \ldots, y^0_j \rangle = \arg\min_{(y_j, \ldots)} E\left( C^0_i \mid x, p^0 \right) \tag{A.2}$$

As the expected value is a linear operator, and after simple transformations, Eq. (A.2) becomes:

$$E\left( C^0_i \mid x, p^0 \right) = 1 - \sum_s \Pr(y_j^0 \mid s) \cdot x \cdot p^0 + \sum_{w=1}^W \sum_{y_{j+1}^0} E\left( C^0_{i+1} \mid x, p^0 \cdot w \right)$$

$$+ E\left( C^0_{i+1} \mid x, p^0 \cdot y_{j+1}^0 \right) \tag{A.3}$$

First, note that the sum of the expected values of $C^0_{i+1}$ and $C^0_i$ in Eq. (A.3) cover all possible suffixes of $p^0$. Hence, they are a constant for every possible value of $y_j$ and the minimization can be done independently. Then,

$$y^0_j = \arg\min_{y_j} \left( 1 - \sum_s \Pr(y_j^0 \mid s) \cdot x \cdot p^0 \right)$$

$$= \arg\max_{y_j} \sum_s \Pr(y_j^0 \mid s) \cdot x \cdot p^0 \tag{A.4}$$

Consequently, $y^0_j$ must form part of the optimum hypothesis. The minimization for subsequent elements can be rewritten as

$$\langle y^0_1, \ldots, y^0_j \rangle = \arg\min_{(y_j, \ldots) \mid y^0_{j+1}} E\left( C^0_i \mid x, p^0 \right) \tag{A.5}$$

Since all but the last term in Eq. (A.3) are constant now that $y^0_j$ has been fixed,

$$\langle y^0_1, \ldots, y^0_j \rangle = \arg\min_{(y_j, \ldots) \mid y^0_{j+1}} E\left( C^0_{i+1} \mid x, p^0 \cdot y^0_j \right) \tag{A.6}$$

Similarly to Eq. (A.4), we can obtain $y^0_{j+1}$. Now, by induction it is trivial to prove that, if Eq. (A.6) holds for $y^0_{j+1}$, it also holds for $y^0_j$, using the same reasoning. That concludes the proof of Proposition 1. □

A.2. Proof of Proposition 2

Proof. For $k = j$, Eqs. (2) and (5) are obviously equivalent since obtaining $y_j$ from Eq. (2),

$$y^0_k = \arg\max_{y_k} \Pr(y_j^0 \mid x, p^0) = \arg\max_{y_j} \Pr(y_j^0 \mid x, p^0) \tag{A.7}$$

for $s = y_{j+1} \ldots, y_j$.

Then, by induction, if both decision rules are equivalent for $y^0_j \cdot k$, then they are also equivalent for $y^0_j \cdot k$,

$$y^0_k = \arg\max_{y_k} \Pr(y_j \mid x, p^0) \Pr(y_k \mid y_{j+1} \cdot k) = \arg\max_{y_k} \Pr(y_k \mid x, p^0) \Pr(y_{j+1} \cdot k \mid x, p^0 \cdot y_j) \tag{A.8}$$

Since $y^0_j \cdot k$ are known to be the optimum values, the first term of the product is constant in Eq. (A.8) finally reaching

$$y^0_k = \arg\max_{y_k} \Pr(y_k \mid y_{j+1} \cdot k \mid x, p^0 \cdot y_j \cdot (k-1)) \tag{A.9}$$

for $s = y_{j+1} \ldots, y_j$. Therefore, both decision rules are equivalent. □

References


Attachment B

Task 2.2

Vicent Alabau, Alberto Sanchis, and Francisco Casacuberta.

Improving on-line handwritten recognition using translation models in multimodal interactive machine translation.

Improving On-line Handwritten Recognition using Translation Models
in Multimodal Interactive Machine Translation

Vicent Alabau, Alberto Sanchis, Francisco Casacuberta
Institut Tecnològic d’Informàtica
Universitat Politècnica de València
Camí de Vera, s/n, Valencia, Spain
{valabau,asanchis,fcn}@iti.upv.es

Abstract

In interactive machine translation (IMT), a human expert is integrated into the core of a machine translation (MT) system. The human expert interacts with the IMT system by partially correcting the errors of the system’s output. Then, the system proposes a new solution. This process is repeated until the output meets the desired quality. In this scenario, the interaction is typically performed using the keyboard and the mouse. In this work, we present an alternative modality to interact within IMT systems by writing on a tactile display or using an electronic pen. An on-line handwritten text recognition (HTR) system has been specifically designed to operate with IMT systems. Our HTR system improves previous approaches in two main aspects. First, HTR decoding is tightly coupled with the IMT system. Second, the language models proposed are context aware, in the sense that they take into account the partial corrections and the source sentence by using a combination of n-grams and word-based IBM models. The proposed system achieves an important boost in performance with respect to previous work.

1 Introduction

Although current state-of-the-art machine translation (MT) systems have improved greatly in the last ten years, they are not able to provide the high quality results that are needed for industrial and business purposes. For that reason, a new interactive paradigm has emerged recently. In interactive machine translation (IMT) (Foster et al., 1998; Barraclough et al., 2009; Koehn and Haddow, 2009) the system goal is not to produce “perfect” translations in a completely automatic way, but to help the user build the translation with the least effort possible.

A typical approach to IMT is shown in Fig. 1. A source sentence $f$ is given to the IMT system. First, the system outputs a translation hypothesis $\hat{e}_s$ in the target language, which would correspond to the output of fully automated MT system. Next, the user analyses the source sentence and the decoded hypothesis, and validates the longest error-free prefix $e_p$ finding the first error. The user, then, corrects the erroneous word by typing some keystrokes $\kappa$, and sends them along with $e_p$ to the system, as a new validated prefix $e_p, \kappa$. With that information, the system is able to produce a new, hopefully improved, suffix $\hat{e}_s$ that continues the previous validated prefix. This process is repeated until the user agrees with the quality of the resulting translation.

Figure 1: Diagram of a typical approach to IMT
narios were proposed, where the user was expected to speak aloud parts of the current hypothesis and possibly one or more corrections. On-line HTR for interactive systems was first explored for interactive transcription of text images (Toselli et al., 2010). Later, we proposed an adaptation to IMT in (Alabau et al., 2010). For both cases, the decoding of the on-line handwritten text is performed independently as a previous step of the suffix $e_s$ decoding. To our knowledge, (Alabau et al., 2010) has been the first and sole approach to the use of on-line handwriting in IMT so far. However, that work did not exploit the specific particularities of the MT scenario.

The novelties of this paper with respect to previous work are summarised in the following items:

- in previous formalisations of the problem, the HTR decoding and the IMT decoding were performed in two steps. Here, a sound statistical formalisation is presented where both systems are tightly coupled.
- the use of specific language modelling for on-line HTR decoding that take into account the previous validated prefix $e_p$, $\kappa$, and the source sentence $f$. A decreasing in error of 2% absolute has been achieved with respect to previous work.
- additionally, a thorough study of the errors committed by the HTR subsystem is presented.

The remainder of this paper is organised as follows: The statistical framework for multimodal IMT and their alternatives will be studied in Sec. 2. Section 3 is devoted to the evaluation of the proposed models. Here, the results will be analysed and compared to previous approaches. Finally, conclusions and future work will be discussed in Sec. 4.

2 Multimodal IMT

In the traditional IMT scenario, the user interacts with the system through a series of corrections introduced with the keyboard. This iterative nature of the process is emphasised by the loop in Fig. 1, which indicates that, for a source sentence to be translated, several interactions between the user and the system should be performed. In each interaction, the system produces the most probable suffix $\hat{e}_s$ that completes the prefix formed by concatenating the longest correct prefix from the previous hypothesis $e_p$ and the keyboard correction $\kappa$. In addition, the concatenation of them, $(e_p, \kappa, \hat{e}_s)$, must be a translation of $f$. Statistically, this problem can be formulated as

$$\hat{e}_s = \arg\max_{e_s} Pr(e_s | e_p, \kappa, f)$$ (1)

The multimodal IMT approach differs from Eq. 1 in that the user introduces the correction using a touch-screen or an electronic pen, $t$. Then, Eq. 1 can be rewritten as

$$\hat{e}_s = \arg\max_{e_s} Pr(e_s | e_p, t, f)$$ (2)

As $t$ is a non-deterministic input (contrarily to $\kappa$), $t$ needs to be decoded in a word $d$ of the vocabulary. Thus, we must marginalise for every possible decoding:

$$\hat{e}_s = \arg\max_{e_s} \sum_d Pr(e_s, d | e_p, t, f)$$ (3)

Furthermore, by applying simple Bayes transformations and making reasonable assumptions,

$$\hat{e}_s \approx \arg\max_{e_s} \frac{Pr(t | d)}{Pr(d | e_p, f)} \cdot Pr(d | e_p, f)$$ (4)

The first term in Eq. 4 is a morphological model and it can be approximated with hidden Markov models (HMM). The last term is an IMT model as described in (Barrachina et al., 2009). Finally, $Pr(d | e_p, f)$ is a constrained language model. Note that the language model is conditioned to the longest correct prefix, just as a regular language model. Besides, it is also conditioned to the source sentence, since $d$ should result of the translation of it.

A typical session of the multimodal IMT is exemplified in Fig. 2. First, the system starts with an empty prefix, so it proposes a full hypothesis. The output would be the same of a fully automated system. Then, the user corrects the first error, $\textit{not}$, by writing $\textit{is}$ on a touch-screen. The HTR subsystem mistakenly recognises $\textit{in}$. Consequently, the user falls back to the keyboard and types $\textit{is}$. Next, the system proposes a new suffix, in which the first word, $\textit{not}$, has been automatically corrected. The user amends $\textit{at}$ by writing the word $\textit{on}$, which is correctly recognised by the HTR subsystem. Finally, as the new proposed suffix is correct, the process ends.
If any feature is not available in your network:

**ITER-0**
- \( (e_p) \)
  - if any feature not is available on your network

**ITER-1**
- \((e_s)\)
- \((e_p)\)
- \((t)\)
- \((d)\)
- \((\kappa)\)
  - \( \kappa \) is
  - \( \in \)
  - \( \notin \)
  - not available at your network

**ITER-2**
- \((e_s)\)
- \((e_p)\)
- \((t)\)
- \((d)\)
  - \( \kappa \) is
  - \( \in \)
  - \( \notin \)
  - not available

**FINAL**
- \((e_s)\)
- \((e_p \equiv e)\)
  - if any feature is not available in your network

**Figure 2:** Example of a multimodal IMT session for translating a Spanish sentence \( f \) from the Xerox corpus to an English sentence \( e \). If the decoding of the pen strokes \( d \) is correct, it is displayed in **boldface**. On the contrary, if \( d \) is incorrect, it is shown **crossed out**. In this case, the user amends the error with the keyboard \( \kappa \) (in **typewriter**).

### 2.1 Decoupled Approach

In (Alabau et al., 2010) we proposed a decoupled approach to Eq. 4, where the on-line HTR decoding was a separate problem from the IMT problem. From Eq. 4 a two step process can be performed. First, \( d \) is obtained,

\[
\hat{d} \approx \underset{d}{\text{argmax}} \ Pr(t|d) \ Pr(d|e_p, f) \tag{5}
\]

Then, the most likely suffix is obtained as in Eq 1, but taking \( \hat{d} \) as the corrected word instead of \( \kappa \),

\[
\hat{e}_s = \underset{e_s}{\text{argmax}} \ Pr(e_s|e_p, \hat{d}, f) \tag{6}
\]

Finally, in that work, the terms of Eq. 5 were interpolated with a unigram in a log-linear model.

### 2.2 Coupled Approach

The formulation presented in Eq. 4 can be tackled directly to perform a coupled decoding. The problem resides in how to model the constrained language model. A first approach is to drop either the \( e_p \) or \( f \) terms from the probability. If \( f \) is dropped, then \( Pr(d|e_p) \) can be modelled as a regular \( n \)-gram model. On the other hand, if \( e_p \) is dropped, but the position of \( d \) in the target sentence \( i = |e_p| + 1 \) is kept, \( Pr(d|f, i) \) can be modelled as a word-based translation model. Let us introduce a hidden variable \( j \) that accounts for a position of a word in \( f \) which is a candidate translation of \( d \). Then,

\[
Pr(d|f, i) = \sum_{j=1}^{|f|} Pr(d, j|f, i) \tag{7}
\]

\[
Pr(d|f, i) \approx \sum_{j=1}^{|f|} Pr(j|f, i)Pr(d|f_j) \tag{8}
\]

Both probabilities, \( Pr(j|f, i) \) and \( Pr(d|f_j) \), can be estimated using IBM models (Brown et al., 1993). The first term is an alignment probability while the second is a word dictionary. Word dictionary probabilities can be directly estimated by IBM1 models. However, word dictionaries are not symmetric. Alternatively, this probability can be estimated using the inverse dictionary to provide a smoothed dictionary,

\[
Pr(d|f_j) = \frac{Pr(d) Pr(f_j|d)}{\sum_{d'} Pr(d') Pr(f_j|d')} \tag{9}
\]

Thus, four word-based translation models have been considered: direct IBM1 and IBM2 models, and inverse IBM1-inv and IBM2-inv models with the inverse dictionary from Eq. 9.

However, a more interesting set up than using language models or translation models alone is to combine both models. Two schemes have been studied.
The most formal under a probabilistic point of view is a linear interpolation of the models,

$$Pr(d|e_p, f) = \alpha Pr(d|e_p) + (1 - \alpha) Pr(d|f, i)$$

(10)

However, a common approach to combine models nowadays is log-linear interpolation (Berger et al., 1996; Papineni et al., 1998; Och and Ney, 2002),

$$Pr(d|e_p, f) = \frac{\exp \left( \sum_m \lambda_m h_m(d, f, e_p) \right)}{Z}$$

(11)

$\lambda_m$ being a scaling factor for model $m$, $h_m$ the log-probability of each model considered in the log-linear interpolation and $Z$ a normalisation factor.

Finally, to balance the absolute values of the morphological model, the constrained language model and the IMT model, these probabilities are combined in a log-linear manner regardless of the language modelling approach.

3 Experiments

The Xerox corpus, created on the TT2 project (SchulmbergerSema S.A. et al., 2001), was used for these experiments, since it has been extensively used in the literature to obtain IMT results. The simplified English and Spanish versions were used to estimate the IMT, IBM and language models. The corpus consists of 56k sentences of training and a development and test sets of 1.1k sentences. Test perplexities for Spanish and English are 33 and 48, respectively.

For on-line HTR, the on-line handwritten UNIPEN corpus (Guyon et al., 1994) was used. The morphological models were represented by continuous density left-to-right character HMMs with Gaussian mixtures, as in speech recognition (Rabiner, 1989), but with variable number of states per character. Feature extraction consisted on speed and size normalisation of pen positions and velocities, resulting in a sequence of vectors of six features (Toselli et al., 2007).

The simulation of user interaction was performed in the following way. First, the publicly available IMT decoder Thot (Ortiz-Martínez et al., 2005) was used to run an off-line simulation for keyboard-based IMT. As a result, a list of words the system failed to predict was obtained. Supposedly, this is the list of words that the user would like to correct with handwriting. Then, from UNIPEN corpus, three users (separated from the training) were selected to simulate user interaction. For each user, the handwritten words were generated by concatenating random character instances from the user’s data to form a single stroke. Finally, the generated handwritten words of the three users were decoded using the corresponding constrained language model with a state-of-the-art HMM decoder, iAtros (Luján-Mares et al., 2008).

Table 1: Comparison of the CER with previous systems. In **boldface** the best system. (†) is an independent, context unaware system used as baseline. (⋆) is a model equivalent to (Alabau et al., 2010).

<table>
<thead>
<tr>
<th>System</th>
<th>Spanish</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev</td>
<td>test</td>
</tr>
<tr>
<td>independent HTR (†)</td>
<td>9.6</td>
<td>10.9</td>
</tr>
<tr>
<td>decoupled (⋆)</td>
<td>9.5</td>
<td>10.8</td>
</tr>
<tr>
<td>best coupled</td>
<td>6.7</td>
<td>8.9</td>
</tr>
</tbody>
</table>

Table 1: Comparison of the CER with previous systems. In **boldface** the best system. (†) is an independent, context unaware system used as baseline. (⋆) is a model equivalent to (Alabau et al., 2010).

3.1 Results

Results are presented in classification error rate (CER), i.e. the ratio between the errors committed by the on-line HTR decoder and the number of handwritten words introduced by the user. All the results have been calculated as the average CER of the three users.

Table 1 shows a comparison between the best results in this work and the approaches in previous work. The log-linear and linear weights were obtained with the simplex algorithm (Nelder and Mead, 1965) to optimise the development set. Then, those weights were used for the test set.

Two baseline models have been established for comparison purposes. On the one hand, (†) is a completely independent and context unaware system. That would be the equivalent to decode the handwritten text in a separate on-line HTR decoder. This system obtains the worst results of all. On the other hand, (⋆) is the most similar model to the best system in (Alabau et al., 2010). This system is clearly outperformed by the proposed coupled approach.

A summary of the alternatives to language mod-
An analysis of the results showed that 52.2% to 61.7% of the recognition errors were produced by punctuation and other symbols. To circumvent this problem, we proposed a contextual menu in (Alabau et al., 2010). With such menu, errors would have been reduced (best test result) to 4.1% in Spanish and 2.8% in English. Out-of-vocabulary (OOV) words also summed up a big percentage of the error (29.1% and 20.4%, respectively). This difference is due to the fact that Spanish is a more inflected language. To solve this problem on-line learning algorithms or methods for dealing with OOV words should be used. Errors in gender, number and verb tenses, which rose up to 7.7% and 5.3% of the errors, could be tackled using linguistic information from both source and target sentences. Finally, the rest of the errors were mostly due to one-to-three letter words, which is basically a problem of handwriting morphological modelling.

4 Conclusions

In this paper we have described a specific on-line HTR system that can serve as an alternative interaction modality to IMT. We have shown that a tight integration of the HTR and IMT decoding process and the use of the available information can produce significant HTR error reductions. Finally, a study of the system’s errors has revealed the system weaknesses, and how they could be addressed in the future.

5 Acknowledgments

Work supported by the EC (FEDER/FSE) and the Spanish MEC/MICINN under the MIPRCV "Consolider Ingenio 2010" program (CSD2007-00018), iTrans2 (TIN2009-14511). Also supported by the Spanish MITyC under the erudito.com (TSI-020110-2009-439) project and by the Generalitat Valenciana under grant Prometeo/2009/014 and GV/2010/067, and by the "Vicerrectorado de Investigación de la UPV" under grant UPV/2009/2851.

References


Attachment C

Task 2.2

Vicent Alabau and Francisco Casacuberta.

Study of electronic pen commands for interactive-predictive machine translation.

In International Workshop on Expertise in Translation and Post-editing Research and Application, 2012.
Study of Electronic Pen Commands for Interactive-Predictive Machine Translation

Vicent Alabau Francisco Casacuberta

Typically, the post-editing of a machine translation (MT) output consists in performing a series of editing operations (i.e., replace, delete, insert or move pieces of text) in a specific text editor using the keyboard and occasionally the mouse. This approach has been proved to be efficient by the translation industry to the point that [1] proposes post-editing guidelines for translation agencies. However, the user needs to be in front of a desktop computer which imposes some restrictions regarding where and how the work is to be done. Laptop computers can also be used, although arguably performance could be diminished because of the use of uncomfortable laptop keyboards and trackpads.

In this work, we envision an alternative scenario in which the user can use a touch screen or an electronic pen (e-pen) to perform post-editing tasks. Although e-pen interaction may sound impractical for texts that need a large amount of post-editing, there is a number of circumstances where it can be more comfortable. First, it can be well suited for post-editing sentences with few errors, as it is the case of sentences with high fuzzy matches, or the revision of human post-edited sentences. Second, it would allow to perform such tasks while commuting, traveling or sitting comfortably on the couch in the living room.

There is already a ‘de facto’ standard for gestures for proof reading (cf. Figure 1) from which we have extracted the most promising gestures: substitutions, deletions, insertions and, transpositions. Furthermore, we have added a shift gesture to move phrases to specific places in the text (i.e., the user circles the phrase and draws an arrow to the final destination). Then, we have studied two e-pen post-editing approaches. In the first one, we consider substitutions, deletions, insertions and, shifts. The number of these operations to obtain a reference can be computed with the translation error rate (TER) [2]. In the second approach, we assume that the user is working with an interactive-predictive MT system (IMT) [3]. In IMT, the user and the MT system collaborate to produce a high-quality output. The user locates the first error from left-to-right and amends it. Then, leveraging the recently validated text, the system reformulates (predicts) the continuation of the translation aiming to improve the previous hypothesis. In this case, we have also considered transpositions.

To know what gestures could be more useful, we have conducted an experiment on the Xerox corpus [4]. The Xerox corpus consists of a collection of technical manuals. It consists of 56k sentences of training and a development and test sets of 1.1k sentences. Test perplexities for Spanish and English are 35 and 51, respectively. The summary of the edit rate results is displayed in Table 1. The edit rate is the number of edit operations needed to obtain the
reference normalized by the number of words. We can see that the IMT system requires less interactions, especially for es-en. Next, the number of times a particular edit operation has been applied is shown. We expect the gestures for deletion, insertion, shifting and transposition to be easy to tell apart for a machine learning algorithm. However, this will be the subject of future work. In addition, substitutions or insertions require the user to write the correct word, which can be done with a virtual keyboard or by handwriting [5]. The perplexities for these words is 336 for English and 242 for Spanish, whereas the errors rates for handwriting recognition are 7.4 for English and 8.9 for Spanish.

References


Table 1: Summary of number of edit operations needed to obtain the reference for post-editing and interactive-predictive machine translation. The edit rate is the ratio between the number of edit operations and the number of words in the reference. Follows the number of occurrences for each edit operation. Here, we assume a perfect gesture recognizer. The gesture recognizer will be developed in future work.

<table>
<thead>
<tr>
<th></th>
<th>post-editing</th>
<th>IMT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>en-es</td>
<td>es-en</td>
</tr>
<tr>
<td>edit rate (%)</td>
<td>21.3</td>
<td>24.4</td>
</tr>
<tr>
<td>substitutions</td>
<td>1028</td>
<td>919</td>
</tr>
<tr>
<td>insertions</td>
<td>325</td>
<td>461</td>
</tr>
<tr>
<td>deletions</td>
<td>484</td>
<td>302</td>
</tr>
<tr>
<td>transpositions</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>shifts</td>
<td>319</td>
<td>357</td>
</tr>
</tbody>
</table>


Attachment D

Task 2.5

Jorge González.

A finite-state approach to phrase-based statistical machine translation.

A finite-state approach to phrase-based statistical machine translation

Jorge González
Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
València, Spain
jgonzalez@dsic.upv.es

Abstract

This paper presents a finite-state approach to phrase-based statistical machine translation where a log-linear modelling framework is implemented by means of an on-the-fly composition of weighted finite-state transducers. Moses, a well-known state-of-the-art system, is used as a machine translation reference in order to validate our results by comparison. Experiments on the TED corpus achieve a similar performance to that yielded by Moses.

1 Introduction

Statistical machine translation (SMT) is a pattern recognition approach to machine translation which was defined by Brown et al. (1993) as follows: given a sentence \( s \) from a certain source language, a corresponding sentence \( t \) in a given target language that maximises the posterior probability \( \Pr(t|s) \) is to be found. State-of-the-art SMT systems model the translation distribution \( \Pr(t|s) \) via the log-linear approach (Och and Ney, 2002):

\[
\hat{t} = \arg\max_t \Pr(t|s) \\
\approx \arg\max_t \sum_{m=1}^{M} \lambda_m h_m(s, t)
\]

where \( h_m(s, t) \) is a logarithmic function representing an important feature for the translation of \( s \) into \( t \), \( M \) is the number of features (or models), and \( \lambda_m \) is the weight of \( h_m \) in the log-linear combination.

This feature set typically includes several translation models so that different relations between a source and a target sentence can be considered. Nowadays, these models are strongly based on phrases, i.e. variable-length \( n \)-grams, which means that they are built from some other lower-context models that, in this case, are defined at phrase level. Phrase-based (PB) models (Tomas and Casacuberta, 2001; Och and Ney, 2002; Marcu and Wong, 2002; Zens et al., 2002) constitute the core of the current state-of-the-art in SMT. The basic idea of PB-SMT systems is:

1. to segment the source sentence into phrases, then
2. to translate each source phrase into a target phrase, and finally
3. to reorder them in order to compose the final translation in the target language.

In a monotone translation framework however, the third step is omitted as the final translation is just generated by concatenation of the target phrases.

Apart from translation functions, the log-linear approach is also usually composed by means of a target language model and some other additional elements such as word penalties or phrase penalties. The word and phrase penalties allow an SMT system to limit the number of words or target phrases, respectively, that constitute a translation hypothesis.

In this paper, a finite-state approach to a PB-SMT state-of-the-art system, Moses (Koehn et al., 2007), is presented. Experimental results validate our work because they are similar to those yielded by Moses. A related study can be found in Kumar et al. (2006) for the alignment template model (Och et al., 1999).
2 Log-linear features for monotone SMT

As a first approach to Moses using finite-state models, a monotone PB-SMT framework is adopted. Under this constraint, Moses’ log-linear model is usually taking into account the following 7 features:

Translation features
1. Direct PB translation probability
2. Inverse PB translation probability
3. Direct PB lexical weighting
4. Inverse PB lexical weighting

Penalty features
5. PB penalty
6. Word penalty

Language features
7. Target language model

2.1 Translation features

All 4 features related to translation are PB models, that is, their associated feature functions \( h_m(s, t) \), which are in any case defined for full sentences, are modelled from other PB distributions \( \eta_m(\tilde{s}, \tilde{t}) \), which are based on phrases.

Direct PB translation probability

The first feature \( h_1(s, t) = \log P(t|s) \) is based on modelling the posterior probability by using the segmentation between \( s \) and \( t \) as a hidden variable \( \beta \). In this manner, \( P(t|s) = \sum_\beta P(t|s, \beta) \) is then approximated by \( P(t|s) \) by using maximization instead of summation, i.e. \( P(t|s) = \max_\beta P(t|s, \beta) \).

Given a monotone segmentation between \( s \) and \( t \), \( P(t|s, \beta) \) is generatively computed as the product of the translation probabilities for each segment pair according to some PB probability distributions:

\[
P(t|s, \beta) = \prod_{k=1}^{K_\beta} P(\tilde{t}_k|\tilde{s}_k)
\]

where \( K_\beta \) is the number of phrases that \( s \) and \( t \) are segmented into, i.e. every \( \tilde{s}_k \) and \( \tilde{t}_k \), respectively, whose dependence on \( \beta \) is omitted for the sake of an easier reading.

Feature 1 is finally formulated as follows:

\[
h_1(s, t) = \log \max_\beta \prod_{k=1}^{K_\beta} P(\tilde{t}_k|\tilde{s}_k) \quad (3)
\]

where \( \eta_1(\tilde{s}, \tilde{t}) = P(\tilde{t}|\tilde{s}) \) is a set of PB probability distributions estimated from bilingual training data, once statistically word-aligned (Brown et al., 1993) by means of GIZA++ (Och and Ney, 2003), which Moses relies on as far as training is concerned. This information is organized as a translation table where a pool of phrase pairs is previously collected.

Inverse PB translation probability

Similar to what happens with Feature 1, Feature 2 is formulated as follows:

\[
h_2(s, t) = \log \max_\beta \prod_{k=1}^{K_\beta} P(\tilde{s}_k|\tilde{t}_k) \quad (4)
\]

where \( \eta_2(\tilde{s}, \tilde{t}) = P(\tilde{s}|\tilde{t}) \) is another set of PB probability distributions, which are simultaneously trained together with the ones for Feature 1, \( P(\tilde{t}|\tilde{s}) \), over the same pool of phrase pairs already extracted.

Direct PB lexical weighting

Given the word-alignments obtained by GIZA++, it is straightforward to estimate a maximum likelihood stochastic dictionary \( P(t_i|s_j) \), which is used to score a weight \( D(\tilde{s}, \tilde{t}) \) to each phrase pair in the pool. Details about the computation of \( D(\tilde{s}, \tilde{t}) \) are given in Koehn et al. (2007). However, as far as this work is concerned, these details are not relevant.

Feature 3 is then similarly formulated as follows:

\[
h_3(s, t) = \log \max_\beta \prod_{k=1}^{K_\beta} D(\tilde{s}_k, \tilde{t}_k) \quad (5)
\]

where \( \eta_3(\tilde{s}, \tilde{t}) = D(\tilde{s}, \tilde{t}) \) is yet another score to use with the pool of phrase pairs aligned during training.

Inverse PB lexical weighting

Similar to what happens with Feature 3, Feature 4 is formulated as follows:

\[
h_4(s, t) = \log \max_\beta \prod_{k=1}^{K_\beta} I(\tilde{s}_k, \tilde{t}_k) \quad (6)
\]
where \( \eta_4(\tilde{s}, \tilde{t}) = I(\tilde{s}, \tilde{t}) \) is another weight vector, which is computed by using a dictionary \( P(s_j|t_i) \), with which the translation table is expanded again, thus scoring a new weight per phrase pair in the pool.

### 2.2 Penalty features

The penalties are not modelled in the same way. The PB penalty is similar to a translation feature, i.e. it is based on a monotone sentence segmentation. The word penalty however is formulated as a whole, being taken into account by Moses at decoding time.

#### PB penalty

The PB penalty scores \( e = 2.718 \) per phrase pair, thus modelling somehow the segmentation length. Therefore, Feature 5 is defined as follows:

\[
h_5(s, t) = \log \max_{\beta} \prod_{k=1}^{K_s} e^{\eta_5(s, t)}
\]

where \( \eta_5(s, t) = e \) extends the PB table once again.

#### Word penalty

Word penalties are not modelled as PB penalties. In fact, this feature is not defined from PB scores, but it is formulated at sentence level just as follows:

\[
h_6(s, t) = \log e^{\|t\|}
\]

where the exponent of \( e \) is the number of words in \( t \).

### 2.3 Language features

Language models approach the a priori probability that a given sentence belongs to a certain language. In SMT, they are usually employed to guarantee that translation hypotheses are built according to the peculiarities of the target language.

#### Target language model

An \( n \)-gram is used as target language model \( P(t) \), where a word-based approach is usually considered. Then, \( h_7(s, t) = \log P(t) \) is based on a model where sentences are generatively built word by word under the influence of the last \( n-1 \) previous words, with the cutoff derived from the start of the sentence:

\[
h_7(s, t) = \log \prod_{i=1}^{\|t\|} P(t_i|t_{i-n+1} \ldots t_{i-1})
\]

where \( P(t_i|t_{i-n+1} \ldots t_{i-1}) \) are word-based probability distributions learnt from monolingual corpora.

### 3 Data structures

This section shows how the features from Section 2 are actually organized into different data structures in order to be efficiently used by the Moses decoder, which implements the search defined by Equation 2 to find out the most likely translation hypothesis \( t \) for a given source sentence \( s \).

#### 3.1 PB models

The PB distributions associated to Features 1 to 5 are organized in table form as a translation table for the collection of phrase pairs previously extracted. That builds a PB database similar to that in Table 1.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>( \eta_1 )</th>
<th>( \eta_2 )</th>
<th>( \eta_3 )</th>
<th>( \eta_4 )</th>
<th>( \eta_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>barato</td>
<td>low cost</td>
<td>1</td>
<td>0.3</td>
<td>1</td>
<td>0.6</td>
<td>2.718</td>
</tr>
<tr>
<td>me gusta</td>
<td>I like</td>
<td>0.6</td>
<td>1</td>
<td>0.9</td>
<td>1</td>
<td>2.718</td>
</tr>
<tr>
<td>es decir</td>
<td>that is</td>
<td>0.8</td>
<td>0.5</td>
<td>0.7</td>
<td>0.9</td>
<td>2.718</td>
</tr>
<tr>
<td>por favor</td>
<td>please</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
<td>0.4</td>
<td>2.718</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>2.718</td>
</tr>
</tbody>
</table>

where each phrase pair is scored by all \( \eta \) models.

#### 3.2 Word-based models

Whereas PB models are an interesting approach to deal with translation relations between languages, language modelling itself is usually based on words. Feature 6 is a length model of the target sentence, and Feature 7 is a target language model.

#### Word penalty

Penalties are not models that need to be trained. However, while PB penalties are provided to Moses to take them into account during the search process (see for example the last column of Table 1, \( \eta_5 \)), word penalties are internally implemented in Moses as part of the log-linear maximization in Equation 2, and are automatically computed on-the-fly at search.

#### Target \( n \)-gram model

Language models, and \( n \)-grams in particular, suffer from a sparseness problem (Rosenfeld, 1996). The \( n \)-gram probability distributions are smoothed to be able to deal with the unseen events out of training data, thus aiming for a larger language coverage.
This smoothing is based on the backoff method, which introduces some penalties for level downgrading within hierarchical language models. For example, let \( \mathcal{M} \) be a trigram language model, which, as regards smoothing, needs both a bigram and a unigram model trained on the same data. Any trigram probability, \( P(c|ab) \), is then computed as follows:

\[
\begin{align*}
\text{if} & \quad abc \in \mathcal{M}: & P_M(c|ab) \\
\text{elseif} & \quad bc \in \mathcal{M}: & BO_M(ab)P_M(c|b) \\
\text{else} & \quad c \in \mathcal{M}: & BO_M(ab)BO_M(b)P_M(unk) \\
\text{else} & & BO_M(ab)BO_M(b)P_M(unk)
\end{align*}
\]

where \( P_M \) is the probability estimated by \( \mathcal{M} \) for the corresponding \( n \)-gram, \( BO_M \) is the backoff weight to deal with the unseen events out of training data, and finally, \( P_M(unk) \) is the probability mass reserved for unknown words.

The \( P(t_{i-1}, t_{i-2}, \ldots, t_0) \) term from Equation 9 is then computed according to that algorithm above, given the model data organized again in table form as a collection of probabilities and backoff weights for the \( n \)-grams appearing in the training corpus. This model displays similarly to that in Table 2.

<table>
<thead>
<tr>
<th>( n )-gram</th>
<th>( P )</th>
<th>BO</th>
</tr>
</thead>
<tbody>
<tr>
<td>please</td>
<td>0.02</td>
<td>0.2</td>
</tr>
<tr>
<td>low cost</td>
<td>0.05</td>
<td>0.3</td>
</tr>
<tr>
<td>I like</td>
<td>0.1</td>
<td>0.7</td>
</tr>
<tr>
<td>that is</td>
<td>0.08</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 2: An English word-based backoff \( n \)-gram model. The likelihood and the backoff model score for each \( n \)-gram.

## 4 Weighted finite-state transducers

Weighted finite-state transducers (Mohri et al., 2002) (WFSTs) are defined by means of a tuple \((\Sigma, \Delta, Q, q_0, f, P)\), where \( \Sigma \) is the alphabet of input symbols, \( \Delta \) is the alphabet of output symbols, \( Q \) is a finite set of states, \( q_0 \in Q \) is the initial state, \( f : Q \rightarrow \mathbb{R} \) is a state-based weight distribution to quantify that states may be final states, and finally, the partial function \( P : Q \times \Sigma^* \times \Delta^* \times Q \rightarrow \mathbb{R} \) defines a set of edges between pairs of states in such a way that every edge is labelled with an input string in \( \Sigma^* \), with an output string in \( \Delta^* \), and is assigned a transition weight.

When weights are probabilities, i.e. the range of functions \( f \) and \( P \) is constrained between 0 and 1, and under certain conditions, a weighted finite-state transducer may define probability distributions. Then, it is called a stochastic finite-state transducer.

### 4.1 WFSTs for SMT models

Here, we show how the SMT models described in Section 3 (that is, the five \( \eta \) scores in the PB translation table, the word penalty, and the \( n \)-gram language model) are represented by means of WFSTs.

First of all, the word penalty feature in Equation 8 is equivalently reformulated as another PB score, as in Equations 3 to 7:

\[
h_6(s, t) = \log e^{|t|} = \log \max_{\beta} \prod_{k=1}^{K_s} e^{|\tilde{t}_k|} \tag{11}
\]

where the length of \( t \) is split up by summation using the length of each phrase in a segmentation \( \beta \). Actually, this feature is independent of \( \beta \), that is, any segmentation produces the expected value \( e^{|t|} \), and therefore the maximization by \( \beta \) is not needed. However, the main goal is to introduce this feature as another PB score similar to those in Features 1 to 5, and so it is redefined following the same framework. The PB table can be now extended by means of \( \eta_6(s, \tilde{t}) = e^{|\tilde{t}|} \), just as Table 3 shows.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>( \eta_1 )</th>
<th>( \eta_2 )</th>
<th>( \eta_3 )</th>
<th>( \eta_4 )</th>
<th>( \eta_5 )</th>
<th>( \eta_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>barato</td>
<td>low cost</td>
<td>...</td>
<td>( e )</td>
<td>( e^2 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>me gusta</td>
<td></td>
<td>...</td>
<td>( e )</td>
<td>( e^2 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>es decir</td>
<td>that is</td>
<td>...</td>
<td>( e )</td>
<td>( e^2 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>por favor</td>
<td>please</td>
<td>...</td>
<td>( e )</td>
<td>( e )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: A word-penalty-extended PB translation table. The exponent of \( e \) in \( \eta_6 \) is the number of words in Target.

Now, the translation table including 6 PB scores and the target-language backoff \( n \)-gram model can be expressed by means of (some stochastic) WFSTs.
Translation table

Each PB model included in the translation table, i.e. any PB distribution in \( \{ \eta_1(\tilde{s}, \tilde{t}), \ldots, \eta_6(\tilde{s}, \tilde{t}) \} \), can be represented as a particular case of a WFST. Figure 1 shows a PB score encoded as a WFST, using a different looping transition per table row within a WFST of only one state.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>( \eta_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>barato / low cost</td>
<td>x_1</td>
<td></td>
</tr>
<tr>
<td>me gusta / I like</td>
<td>x_2</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Equivalent WFST representation of PB scores. Table rows are embedded within as many looping transitions of a WFST which has no topology at all; \( \eta \)-scores are correspondingly stored as transition weights.

Language model

It is well known that \( n \)-gram models are a subclass of stochastic finite-state automata where backoff can also be adequately incorporated (Llorens, 2000).

Then, they can be equivalently turned into transducers by means of the concept of identity, that is, transducers which map every input label to itself. Figure 2 shows a WFST for a backoff bigram model.

It is also quite straight-forward to see that the application of the Viterbi method (Viterbi, 1967) on these WFSTs provides the corresponding feature value \( h_m(s, t) \) for all Features 1 to 6 as defined in Equations 3 to 8.

4.2 Search

Equation 2 is a general framework for log-linear approaches to SMT. This framework is adopted here in order to combine several features based on WFSTs, which are modelled as their respective Viterbi score.

As already mentioned, the computation of \( h_m(s, t) \) for each PB-WFST, let us say \( T_m \) (with \( 1 \leq m \leq 6 \)), provides the most likely segmentation \( \beta_m \) for \( s \) and \( t \) according to \( T_m \). However, a constraint is used here so that all \( T_m \) models define the same segmentation:

\[
K_{\beta} > 0
\]

\[
s = \tilde{s}_1 \ldots \tilde{s}_{K_{\beta}}
\]

\[
t = \tilde{t}_1 \ldots \tilde{t}_{K_{\beta}}
\]

where the PB scores corresponding to Features 1 to 6 are directly applied on that particular segmentation for each phrase pair \((\tilde{s}_k, \tilde{t}_k)\) monotonically aligned. Equations 3 to 7 and 11 can be simplified as follows:

\[
\forall m = 1, \ldots, 6
\]

\[
h_m(s, t) = \log \prod_{k=1}^{K_{\beta}} \eta_m(\tilde{s}_k, \tilde{t}_k) \quad (12)
\]

To sum up, our log-linear combination scenario considers 7 (some stochastic) WFSTs, 1 per feature: 6 of them are PB models related to a translation table while the 7th one is a target-language \( n \)-gram model.

Next in Section 4.2, we show how these WFSTs are used in conjunction in a homogeneous framework.
Then, Equation 2 can be instanced as follows:

\[
\hat{t} = \arg \max_t \sum_{m=1}^{7} \lambda_m h_m(s, t) \tag{13}
\]

\[
= \arg \max_t \sum_{m=1}^{7} \lambda_m \sum_{k=1}^{K_{\theta}} \log \eta_m(\tilde{s}_k, \tilde{t}_k) + \lambda_{\gamma} \sum_{i=1}^{\|t\|} \log P(t_i|t_{i-n+1} \ldots t_{i-1})
\]

\[
= \arg \max_t \sum_{m=1}^{7} \lambda_m \sum_{k=1}^{K_{\theta}} \log \eta_m(\tilde{s}_k, \tilde{t}_k) + \lambda_{\gamma} \sum_{i=1}^{\|t\|} \log P(t_i|t_{i-n+1} \ldots t_{i-1})
\]

as logarithm rules are applied to Equations 9 and 12.

The square-bracketed expression of Equation 13 is a Viterbi-like score which can be incrementally built through the contribution of all the PB-WFSTs (along with their respective \(\lambda_m\)-weights) over some phrase pair \((\tilde{s}_k, \tilde{t}_k)\) that extends a partial hypothesis. As these models share their topology, we implement them jointly including as many scores per transition as needed (González and Casacuberta, 2008). These models can also be merged by means of union once their \(\lambda_m\)-weights are transferred into them. That allows us to model the whole translation table (see Table 3) by means of just 1 WFST structure \(T\). Therefore, the search framework for single models can also be used for their log-linear combination.

As regards the remaining term from Equation 13, i.e. the target \(n\)-gram language model for Feature 7, it is seen as a rescoring function (Och et al., 2004) which is applied once the PB-WFST \(T\) is explored. The translation model returns the best hypotheses that are later input to the \(n\)-gram language model \(L\), where they are reranked, to finally choose the best \(\hat{t}\).

However, these two steps can be processed at once if both the WFST \(T\) and the WFST \(L\) are merged by means of their composition \(T \circ L\) (Mohri, 2004). The product of such an operation is another WFST as WFSTs are closed under a composition operation. In practice though, the size of \(T \circ L\) can be very large so composition is done on-the-fly (Caseiro, 2003), which actually does not build the WFST for \(T \circ L\) but explores both \(T\) and \(L\) as if they were composed, using the \(n\)-gram scores in \(L\) on the target hypotheses from \(T\) as soon as they are partially produced.

Equation 13 represents a Viterbi-based composition framework where all the (weighted) models contribute to the overall score to be maximized, provided that the set of \(\lambda_m\)-weights is instantiated. Using a development corpus, the set of \(\lambda_m\)-weights can be empirically determined by means of running several iterations of this framework, where different values for the \(\lambda_m\)-weights are tried in each iteration.

5 Experiments

Experiments were carried out on the TED corpus, which is described in depth throughout Section 5.1. Automatic evaluation for SMT is often considered and we use the measures enumerated in Section 5.2. Results are shown and also discussed, in Section 5.3.

5.1 Corpora data

The TED corpus is composed of a collection of English-French sentences from audiovisual content whose main statistics are displayed in Table 4.

<table>
<thead>
<tr>
<th>Subset</th>
<th>English</th>
<th>French</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td>47.5K</td>
<td>792.9K</td>
</tr>
<tr>
<td>Running words</td>
<td>747.2K</td>
<td>31.7K</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>24.6K</td>
<td>31.7K</td>
</tr>
<tr>
<td>Develop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td>571</td>
<td></td>
</tr>
<tr>
<td>Running words</td>
<td>9.2K</td>
<td>10.3K</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>1.9K</td>
<td>2.2K</td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td>641</td>
<td></td>
</tr>
<tr>
<td>Running words</td>
<td>12.6K</td>
<td>12.8K</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>2.4K</td>
<td>2.7K</td>
</tr>
</tbody>
</table>

Table 4: Main statistics from the TED corpus and its split.

As shown in Table 4, develop and test partitions are statistically comparable. The former is used to train the \(\lambda_m\)-weights in the log-linear approach, in the hope that they can also work well for the latter.

5.2 Evaluation measures

Since its appearance as a translation quality measure, the BLEU metric (Papineni et al., 2002), which stands for bilingual evaluation understudy, has become consolidated in the area of automatic evaluation as the most widely used SMT measure. Nevertheless, it was later found that its correlation factor...
with subjective evaluations (the original reason for its success) is actually not so high as first thought (Callison-Burch et al., 2006). Anyway, it is still the most popular SMT measure in the literature.

However, the word error rate (WER) is a very common measure in the area of speech recognition which is also quite usually applied in SMT (Och et al., 1999). Although it is not so widely employed as BLEU, there exists some work that shows a better correlation of WER with human assessments (Paul et al., 2007). Of course, the WER measure has some bad reviews as well (Chen and Goodman, 1996; Wang et al., 2003) and one of the main criticisms that it receives in SMT areas is about the fact that there is only one translation reference to compare with. The MWER measure (Nießen et al., 2000) is an attempt to relax this dependence by means of an average error rate with respect to a set of multiple references of equivalent meaning, provided that they are available.

Another measure also based on the edit distance concept has recently arisen as an evolution of WER towards SMT. It is the translation edit rate (TER), and it has become popular because it takes into account the basic post-process operations that professional translators usually do during their daily work. Statistically, it is considered as a measure highly correlated with the result of one or more subjective evaluations (Snover et al., 2006).

The definition of these evaluation measures is as follows:

**BLEU**: It computes the precision of the unigrams, bigrams, trigrams, and fourgrams that appear in the hypotheses with respect to the \( n \)-grams of the same order that occur in the translation reference, with a penalty for too short sentences. Unlike the WER measure, BLEU is not an error rate but an accuracy measure.

**WER**: This measure computes the minimum number of editions (replacements, insertions or deletions) that are needed to turn the system hypothesis into the corresponding reference.

**TER**: It is computed similarly to WER, using an additional edit operation. TER allows the movement of phrases, besides replacements, insertions, and deletions.

### 5.3 Results

The goal of this section is to assess experimentally the finite-state approach to PB-SMT presented here. First, an English-to-French translation is considered, then a French-to-English direction is later evaluated.

On the one hand, our log-linear framework is tuned on the basis of BLEU as the only evaluation measure in order to select the best set of \( \lambda_m \)-weights. That is accomplished by means of development data, however, once the \( \lambda_m \)-weights are estimated, they are extrapolated to test data for the final evaluation.

Table 5 shows: a) the BLEU translation results for the development data; and b) the BLEU, WER and TER results for the test data. In both a) and b), the \( \lambda_m \)-weights are trained on the development partition. These results are according to different feature combinations in our log-linear approach to PB-SMT.

As shown in Table 5, the first experimental scenario is not a log-linear framework since only one feature, (a direct PB translation probability model) is considered. The corresponding results are poor and, judging by the remaining results in Table 5, they reflect the need for a log-linear approach.

The following experiments in Table 5 represent a log-linear framework for Features 1 to 6, i.e. the PB translation table encoded as a WFST \( T \), where different PB models are the focus of attention. Only the log-linear combination of Features 1 and 2

<table>
<thead>
<tr>
<th>Log-linear features</th>
<th>Develop BLEU</th>
<th>Test BLEU</th>
<th>Test WER</th>
<th>Test TER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (baseline)</td>
<td>8.5</td>
<td>7.1</td>
<td>102.9</td>
<td>101.5</td>
</tr>
<tr>
<td>1+2</td>
<td>4.0</td>
<td>3.0</td>
<td>116.6</td>
<td>115.6</td>
</tr>
<tr>
<td>1+2+3</td>
<td>22.7</td>
<td>18.4</td>
<td>66.6</td>
<td>64.4</td>
</tr>
<tr>
<td>1+2+3+4</td>
<td>22.8</td>
<td>18.5</td>
<td>66.3</td>
<td>64.2</td>
</tr>
<tr>
<td>1+2+3+4+5</td>
<td>22.7</td>
<td>18.8</td>
<td>65.2</td>
<td>63.2</td>
</tr>
<tr>
<td>1+2+3+4+5+6</td>
<td>23.1</td>
<td>19.1</td>
<td>65.9</td>
<td>63.8</td>
</tr>
<tr>
<td>1+7</td>
<td>24.6</td>
<td>20.5</td>
<td>65.1</td>
<td>62.9</td>
</tr>
<tr>
<td>1+2+7</td>
<td>25.5</td>
<td>21.3</td>
<td>63.7</td>
<td>61.6</td>
</tr>
<tr>
<td>1+2+3+7</td>
<td>25.9</td>
<td>22.2</td>
<td>62.5</td>
<td>60.4</td>
</tr>
<tr>
<td>1+2+3+4+7</td>
<td>26.3</td>
<td>22.0</td>
<td>63.4</td>
<td>61.3</td>
</tr>
<tr>
<td>1+2+3+4+5+7</td>
<td>26.4</td>
<td>22.1</td>
<td>63.1</td>
<td>61.0</td>
</tr>
<tr>
<td>1+2+3+4+5+6+7</td>
<td>27.0</td>
<td>21.8</td>
<td>64.4</td>
<td>62.2</td>
</tr>
<tr>
<td>Moses (1+...+7)</td>
<td>27.1</td>
<td>22.0</td>
<td>64.0</td>
<td>61.8</td>
</tr>
</tbody>
</table>

Table 5: English-to-French results for development and test data according to different log-linear scenarios. The set of \( \lambda_m \)-weights is learnt from development data for every feature combination log-linear scenario defined.
is worse than the baseline, which feeds us back on the fact that the $\lambda_m$-weights can be better trained, that is, the log-linear model for Features 1 and 2 can be upgraded until baseline’s results with $\lambda_2 = 0$. This battery of experiments on Features 1 to 6 allows us to see the benefits of a log-linear approach. The baseline results are clearly outperformed now, and we can say that the more features are included, the better are the results.

The next block of experiments in Table 5 always include Feature 7, i.e. the target language model $T$. Features 1 to 6 are progressively introduced into $T$. These results confirm that the target language model is still an important feature to take into account, even though PB models are already providing a surrounding context for their translation hypotheses because translation itself is modelled at phrase level. These results are significantly better than the ones where the target language model is not considered. Again, the more translation features are included, the better are the results on the development data. However, an overtraining is presumably occurring with regard to the optimization of the $\lambda_m$-weights, as results on the test partition do not reach their top the same way the ones for the development data do, i.e. when using all 7 features, but when combining Features 1, 2, 3, and 7, instead. These differences are not statistically significant though.

Finally, our finite-state approach to PB-SMT is validated by comparison, as it allows us to achieve similar results to those yielded by Moses itself.

On the other hand, a translation direction where French is translated into English gets now the focus. Their corresponding results are presented in Table 6. A similar behaviour can be observed in Table 6 for the series of French-to-English empirical results.

<table>
<thead>
<tr>
<th>Log-linear features</th>
<th>Develop BLEU</th>
<th>Test BLEU</th>
<th>WER</th>
<th>TER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (baseline)</td>
<td>7.1</td>
<td>101.6</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>1+2</td>
<td>4.1</td>
<td>117.5</td>
<td>116.0</td>
<td></td>
</tr>
<tr>
<td>1+2+3</td>
<td>24.2</td>
<td>59.8</td>
<td>56.5</td>
<td></td>
</tr>
<tr>
<td>1+2+3+4</td>
<td>24.4</td>
<td>58.0</td>
<td>55.7</td>
<td></td>
</tr>
<tr>
<td>1+2+3+4+5</td>
<td>24.9</td>
<td>56.9</td>
<td>54.8</td>
<td></td>
</tr>
<tr>
<td>1+2+3+4+5+6</td>
<td>25.2</td>
<td>57.1</td>
<td>55.0</td>
<td></td>
</tr>
<tr>
<td>1+7</td>
<td>24.7</td>
<td>60.0</td>
<td>57.7</td>
<td></td>
</tr>
<tr>
<td>1+2+7</td>
<td>26.0</td>
<td>58.8</td>
<td>56.5</td>
<td></td>
</tr>
<tr>
<td>1+2+3+7</td>
<td>28.5</td>
<td>56.1</td>
<td>54.0</td>
<td></td>
</tr>
<tr>
<td>1+2+3+4+7</td>
<td>28.4</td>
<td>56.0</td>
<td>53.8</td>
<td></td>
</tr>
<tr>
<td>1+2+3+4+5+7</td>
<td>28.8</td>
<td>56.0</td>
<td>53.9</td>
<td></td>
</tr>
<tr>
<td>1+2+3+4+5+6+7</td>
<td>28.7</td>
<td>55.8</td>
<td>53.7</td>
<td></td>
</tr>
<tr>
<td>Moses (1+...+7)</td>
<td>28.9</td>
<td>55.8</td>
<td>53.6</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: French-to-English results for development and test data according to different log-linear scenarios.

These models can also be implemented by means of WFSTs on the basis of the Viterbi algorithm. The word penalty can also be equivalently redefined as another PB model, similar to the five others, which allows us to constitute a translation model $T$ composed of six parallel WFSTs that are constrained to share the same monotonic bilingual segmentation.

A backoff $n$-gram model for the target language $L$ can be represented as an identity WFST where $P(t)$ is modelled on the basis of the Viterbi algorithm. The whole log-linear approach to Moses is attained by means of the on-the-fly WFST composition $T \circ L$. Our finite-state log-linear approach to PB-SMT is validated by comparison, as it has allowed us to achieve similar results to those yielded by Moses.

Monotonicity is an evident limitation of this work, as Moses can also feature some limited reordering. However, future work on that line is straightforward since the framework described in this paper can be easily extended to include a PB reordering model $R$, by means of the on-the-fly composition $T \circ R \circ L$.

6 Conclusions and future work

In this paper, a finite-state approach to Moses, which is a PB-SMT state-of-the-art system, is presented. A monotone framework is adopted, where 7 models in log-linear combination are considered: a direct and an inverse PB translation probability model, a direct and an inverse PB lexical weighting model, PB and word penalties, and a target language model.

Five out of these models are based on PB scores which are organized under a PB translation table.
References


