D2.2: Progress Report on Interactive Translation Prediction

Germán Sanchis-Trilles, Vicent Alabau, Francisco Casacuberta, José-Miguel Benedí, Jesús González-Rubio, Daniel Ortiz-Martínez, Philipp Koehn

Distribution: Public

CASMACAT
Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation
ICT Project 287576 Deliverable D2.2

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

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contractual date of delivery</td>
<td>October 31, 2013</td>
</tr>
<tr>
<td>Actual date of delivery</td>
<td></td>
</tr>
<tr>
<td>Date of last update</td>
<td>November 5, 2013</td>
</tr>
<tr>
<td>Deliverable number</td>
<td>D2.2</td>
</tr>
<tr>
<td>Deliverable title</td>
<td>Progress Report on Interactive Translation Prediction</td>
</tr>
<tr>
<td>Type</td>
<td>Report</td>
</tr>
<tr>
<td>Status &amp; version</td>
<td>Final</td>
</tr>
<tr>
<td>Number of pages</td>
<td>96</td>
</tr>
<tr>
<td>Contributing WP(s)</td>
<td>WP2</td>
</tr>
<tr>
<td>WP / Task responsible</td>
<td>UPVLC</td>
</tr>
<tr>
<td>Other contributors</td>
<td>CS</td>
</tr>
<tr>
<td>Internal reviewer</td>
<td>Michael Carl</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Germán Sanchis-Trilles, Vicent Alabau, Francisco Casacuberta, José-Miguel Benedí, Jesús González-Rubio, Daniel Ortiz-Martínez, Philipp Koehn</td>
</tr>
<tr>
<td>EC project officer</td>
<td>Kimmo Rossi</td>
</tr>
<tr>
<td>Keywords</td>
<td></td>
</tr>
</tbody>
</table>

The partners in CASMACAT are:

University of Edinburgh (UEDIN)
Copenhagen Business School (CBS)
Universitat Politècnica de València (UPVLC)
Celer Soluciones (CS)

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

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

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

© 2013, 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 second year of WP2, Interactive Translation Prediction, of the CASMACAT project. This work-package has been one of the most active ones during the second year, with four of its five tasks 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 new search strategies, novel ways of user interaction, or less conventional SMT models being applied to ITP. The main goal of this work-package is to supply the final user with a more flexible and efficient assistance system, be it by means of multi-modality, by introducing novel ways of interacting with the system, or by improving the quality of the translation completions provided by the underlying ITP system.

During this second year, work was performed in tasks 2.1, 2.2, 2.3, and 2.4:

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 optimum approach, motivated by the Bayesian decision theory. During the last period of this task, a greedy version of the optimum prediction algorithm has been developed, leading to a reduction in the computational cost of the original optimal algorithm and a reduction in the size of the word-graphs used by the ITP system. The first part of this work has been published in an important international journal. In the second place, the error-correcting technology developed during the first year of this task was incorporated into the CASMACAT prototype and is now being used in all the experiments.

Task 2.2 Multi-modality in Interactive Translation Prediction (month 6-24)
Work in this direction focuses on using an e-pen interface for interacting with the post-editing or ITP system. It allows correcting the hypotheses by just hand writing and/or by performing simple gestures with an e-pen. The e-pen interface aims at increasing the interacting ergonomy and naturalness, as well as at enabling ITP or post-editing in more casual setting than those based on the conventional computers with keyboard interfaces. However, no e-pen interface can ever be error-free and the challenge is to take advantage of information derived from the ITP process to boost the e-pen accuracy and robustness. The work conducted so far has lead to several publications in relevant conferences. Work currently in progress mainly aims at fully allowing both character-level and multi-word corrective interactions, since only single-word corrections are allowed in the version currently being implemented into the CASMACAT workbench.

Task 2.3 Prediction from active interaction (month 13-36)
The work performed in this task concerns providing the human translator with a tool by means of which he can spot faster which parts of the translation suggestions provided by the underlying SMT system are prone to contain errors. For doing this, confidence measures are used, so as to display potentially incorrect words in a different colour. So far, the work performed during the first year of activity of this task has focused on providing an initial implementation of confidence measures within the CASMACAT framework. Such implementation was evaluated by users in the second field trial, revealing that, even though users seem to appreciate such a tool, further research is still needed.

Task 2.4 Prediction from Parse Forest (month 6-18) The work conducted in the last period of this task has lead to a fully-functional grammar-based ITP system. Even though not tested in field trials, the laboratory results obtained are very appealing, and have been published in an important international conference.
Contents

1 Executive Summary .............................................. 3

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

3 Task 1: Search and machine learning criteria for prediction .... 7
   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 ........................................... 9
   3.2 ITP Based on Stochastic Error-Correction Models ............... 9
   3.3 Prediction as a Machine Learning Problem .................. 11
      3.3.1 Evaluation and Baseline ............................. 11
      3.3.2 Absolute Maximum Accuracy of Word Predictions ........ 12
      3.3.3 Machine Learning .................................... 12
      3.3.4 Exploring features using SVM ........................ 13
      3.3.5 Evaluation of the Extended Model .................... 13
      3.3.6 Conclusion ........................................... 13

4 Task 2: Multi-modality in interactive translation prediction .... 13
   4.1 Introduction ................................................. 13
   4.2 Fundamentals of multi-modal interactive text editing .......... 14
   4.3 Interactive text transcription using e-pen feedback ........ 15
   4.4 Robust on-line HTR by leveraging MT models ............... 15
   4.5 Study of e-pen gestures for ITP .......................... 15
   4.6 Conclusions ................................................ 15

5 Task 3: Prediction from Active Interaction ...................... 16
   5.1 Introduction ................................................ 16
   5.2 Development .............................................. 16
   5.3 Results .................................................... 17
   5.4 Conclusions ............................................... 20

6 Task 4: Prediction from Parse Forest ............................ 20
   6.1 Introduction ................................................ 20
   6.2 ITP using Hierarchical translation Models .................. 20
      6.2.1 Introduction ........................................... 20
      6.2.2 Development ........................................... 21
      6.2.3 Conclusions ........................................... 21
   6.3 Approximate Search for User Prefix ........................ 21
      6.3.1 Introduction ........................................... 21
      6.3.2 Summary of Methods .................................. 21
      6.3.3 Conclusions ........................................... 22

Bibliography ..................................................... 23

Attachment A .................................................... 27

Attachment B .................................................... 35

Attachment C .................................................... 63
2 Background

Despite multiple and important advances obtained so far in the field of Statistical Machine Translation (SMT), current machine translation (MT) systems are in many cases not able to produce ready-to-use texts \[1, 9\]. 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 \[16\]. 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 \[25\] 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 \(x_J^j \equiv x_1...x_j...x_J\) in the source language \(\mathcal{X}\), we want to find its equivalent target sentence \(y_I^i \equiv y_1...y_i...y_I\) in the target language \(\mathcal{Y}\).

\(^1_x_j\) and \(y_i\) note the \(i^{th}\) word and the \(j^{th}\) word of the sentences \(x_J^j\) and \(y_I^i\) respectively.
From the set of all possible sentences of the target language, we are interested in the one with the highest probability according to the following equation:

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

(1)

where $Pr(y_1^T \mid x_1^T)$ represents the probability of translating $x_1^T$ into $y_1^T$.

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 $Pr(y_1^T \mid x_1^T)$ applying the Bayes decision rule. Taking into account that $Pr(x_1^T)$ does not depend on $y_1^T$ we arrive to the following expression [7]:

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

(2)

where: $Pr(y_1^T)$ represents the probability of generating the target sentence, and $Pr(x_1^T \mid y_1^T)$ is the probability of generating $y_1^T$ given $x_1^T$. Since the real probability distributions $Pr(y_1^T)$ and $Pr(x_1^T \mid y_1^T)$ are not known, they are approximated by means of parametric statistical models. Specifically, $Pr(y_1^T)$ is modelled by means of a language model, and $Pr(x_1^T \mid y_1^T)$ is modelled by means of a translation model. Current MT systems are based on the use of phrase-based models [17] 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 $Pr(y_1^T \mid x_1^T)$, replacing the generative models by discriminative models. Log-linear models use a set of feature functions $h_m(x_1^T,y_1^T)$ each one with its corresponding weight $\lambda_m$:

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

(3)

The direct optimisation of the posterior probability in the Bayes decision rule is referred to as discriminative training [24]. 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 [26].

### 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 (referred to from now on as 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:

$$\hat{y}_s = \arg \max_{y_s} \{ p(y_s \mid x_1^T, y_p) \}$$

(4)

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

$$\hat{y}_s = \arg \max_{y_s} \{ p(y_s \mid y_p) \cdot p(x_1^T \mid y_p, y_s) \}$$

(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 \( y_I^J \) which contain \( y_p \) as prefix. It is also worth mentioning that the similarities between Equation (4) and Equation (2) (note that \( y_p y_s \equiv y_I^J \)) allow us to use the same models if the search procedures are adequately modified \([5, 4]\).

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 \([Equation 5]\). 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 deliverable D2.1, we presented an optimal decision rule for ITP as a result of the work presented in \([2]\). In summary, the suffix is built by concatenating words, one word at a time, until the end of sentence symbol is predicted. Suppose that we have a prefix of length \( i \), and we are predicting the word in the \( k \)-th position, then, the optimum decision rule in \([2]\) states that:

\[
\hat{y}_k = \arg \max_{y_k} \sum_{y'_s} p(y_k; y'_s|x_1^J, y_p; \hat{y}_{k-1}^{i+1})
\]  

(6)

where \( \cdot \) is the concatenation operator. It should be noted that the probability in \([Equation 6]\) \( p(y_k; y'_s|x_1^J, y_p; \hat{y}_{k-1}^{i+1}) \), is an instantiation of the probability in \([Equation 4]\). Furthermore, also note that the complete prediction of the suffix \( y_s \) is obtained by applying \([Equation 6]\) repeatedly.

In addition, \([2]\) proves that \([Equation 4]\) can also be computed as:

\[
\hat{y}_k = \arg \max_{y_k} \max_{y'_s} p(y_k; y'_s|x_1^J, y_p; \hat{y}_{k-1}^{i+1})
\]  

(7)

which is an approximation to \([Equation 6]\) where the sum is replaced its maximum addend.

A practical and efficient algorithm for these equations that operates on word graphs (WG) was proposed in \([2]\). Formally, a WG is a directed, acyclic, weighted graph generated from \( x_1^J \) and defined by the tuple \( G(x_1^J) = (Q, q_I, q_F, \mathcal{Y}, A, \mathcal{F}) \) where:

- \( Q \) is a set of nodes being \( q_I \) the initial node and \( q_F \) the final node.
- \( \mathcal{Y} \) is the set of possible output labels.
- \( A : \mathcal{Y} \times Q \times Q \) is a set of arcs, \( e = (y, u, v) \), where \( y \) is an output label that is generated from a start node \( u \) to an end node \( v \).
- \( \mathcal{F}(e) : A \rightarrow \mathcal{R} \) is a score function that evaluates how likely is \( y \) to be generated in the context of nodes \( u \) and \( v \).
As explained in [2], given a WG, the following equation can be used to approximate the posterior probability in Equation 6:

\[
\hat{y} = \arg \max_y \sum_{y'} p(y; y'|x_1^I, y_p) \simeq \arg \max_y \sum_u \sum_v p((y, u, v)|x_1^I, y_p) \quad (8)
\]

with

\[
p((y, u, v)|x_1^I, y_p) = \Phi_{y_p}(u)\mathcal{F}((y, u, v))\Psi(v) \sum_{u'} \Phi_{y_p}(u')\Psi(u') \quad (9)
\]

where \(\Phi_{y_p}(u)\) is the forward probability only taking into account the paths that match \(y_p\) and \(\Psi(v)\) is the backward probability [37].

If the forward probability is replaced by the probability of the maximum prefix, and the backward probability is replaced also by the most likely suffix, then Equation 8 computes Equation 7. Thus, we can conclude that the classical algorithm is a maximum approximation to the optimum algorithm. In terms of weighted finite-state automata (WFA) theory [23], when the scores in the WG are log-likelihoods, we could say that the weights of the classical algorithm have the structure of a tropical semiring, whereas the weights of the optimal algorithm have the structure of a log semiring. Consequently, we operate on the WGs to obtain the classical or the optimum approach just by using the same WFA algorithms with the corresponding structure of semirings.

A greedy version of the optimum algorithm

Although Equation 8 is an efficient algorithm to compute Equation 6, it is still necessary to sum over all possible nodes \(u\) and \(v\) for each of the predicted words in the suffix.

Here, we present a transformation of the WG using WFA tools after which the optimum algorithm becomes a greedy algorithm, in the sense that decisions can be taken locally. In consequence, after the transformation, the optimum algorithm runs in linear time in the length of the suffix.

First, we define the normalisation of the WG into a probabilistic automaton [31] in similar terms to Equation 9 as:

\[
p((y, u, v)|x_1^I) = \frac{\mathcal{F}((y, u, v))\Psi(v)}{\Psi(u)} \quad (10)
\]

Proposition 1. If the word graph is a deterministic probabilistic WFA then a greedy algorithm is optimum w.r.t. the cost of interaction.

Proof. First, note that if the WG is deterministic, the double summation in Equation 8 is made over a single element, since by definition of determinism from a node only one arc can exist with the same label. Consequently, there is a unique \(u\) where the prefix \(y_p\) ends. Thus, the normalisation factor can be expressed as \(\Phi_{y_p}(u)\Psi(u)\). Then,

\[
\sum_{y'} p(y; y'|x_1^I, y_p) \simeq p((y, u, v)|x_1^I, y_p)
\]

\[
= \Phi_{y_p}(u)\mathcal{F}(e)\Psi(v) \quad (12)
\]

\[
= \mathcal{F}(e)\Psi(v) \quad (13)
\]

which is the normalisation solution in Equation 10. \(\Box\)

\[\text{The classical algorithm can also be greedy if the tropical semiring is used.}\]
In summary, by normalising and determinising the WG the optimum algorithm becomes greedy. Note that the general algorithm for automata determinisation does not guarantee termination. However, any acyclic weighted automaton over a zero-sum-free semiring is determinisable \[23\]. Fortunately, WGs are such kind of automata, either they work in the \emph{tropical semiring} or the \emph{log semiring}. However, the worst-case scenario of determinisation is exponential in time, and the resulting WG may also grow exponentially in size, which in part depends on the type of semiring and the number of bits necessary to represent the weights \[8\]. Nevertheless, the general consensus is that WGs produced by i.e. speech recognition systems can be determinised fast, and a further minimization of the determinised WG can result in a WG smaller than the original one.

Figure 1 is a study of the size of the transformed WG for different kinds of weights for the \texttt{newstest2011} corpus used in the \textsc{casmacat} evaluation. \texttt{unweighted} stands for a WG whose scores have been removed, whereas \texttt{log semiring} corresponds to the optimum approach and \texttt{tropical semiring} to the classical approach.

If we analyse the \texttt{log semiring} plots, we can see that the number of states can be up to 2000 times of original WG. Arcs increase even more dramatically, which suggest an exponential state expansion in the determinisation algorithm. The results of the \texttt{tropical semiring} are one order of magnitude less, probably as a result of using the maximum instead of the sum. Finally, the results show that by determinising and minimizing unweighted WGs, the resulting WG is reduced typically to \(\simeq 20\%\) of the original number of states and to \(\simeq 50\%\) of the original number of arcs, which seems a huge saving. In addition, the transformation is very fast, taking less than 500\,ms in most of the cases. However, this kind of WGs is useless for our purpose since these WGs do not provide probability information and cannot be used for decoding. Therefore, a possible solution is to restore the probabilities of Equation 8 in the original WG into the minimized unweighted WG as Equation 13. Although we have already found an algorithm for this, the proof of correctness still needs some revision.

### 3.1.3 Conclusions

We have shown that WG determinisation and minimization can transform the optimum algorithm for ITP in a greedy version of the algorithm that runs linear in time in the length of the suffix. However, current algorithms for WG determinisation and minimization present an exponential growth in the number of states and arcs that make the transformation impractical for MT WGs. Fortunately, we also have observed that when determinisation and minimization are performed on the unweighted WGs, the number of states in the WG can be reduced to a fifth and the number of arcs can be halved. This, together with the reduction in the complexity of the algorithm may lead to efficiency gains in the \textsc{casmacat} prototype. The formal proof thereof should be available in the next few months, and once this is done it would be possible to incorporate such technology into the prototype.

### 3.2 ITP Based on Stochastic Error-Correction Models

Error-correction techniques play an important role in existing word-graph based ITP systems to treat those situations in which the prefix given by the user cannot be explained by the statistical models \[4\]. However, such techniques are not included in the statistical formulation of the ITP process, and thus they are used in a heuristic manner. In the initial year of project, we described the statistical foundations of an ITP system that successfully includes stochastic error-correction models in its formulation.

During the second year of the project, the work on ITP based on stochastic error-correction models has been focused on two different directions, namely, possible extensions of the proposed
unweighted

Figure 1: Performance of WG determinisation and minimization for different kinds of weights for a set of 100 random WGs in the newstest2011 corpus. unweighted stands for a WG whose probabilities have been removed, whereas log semiring sums the weights of the paths and tropical semiring uses the maximum approach. Note that the log semiring is the one required for the optimum approach.
formalism and practical implementation and deployment of an ITP system following such formalism.

Regarding the extensions to the statistical ITP formalism proposed in the first year of the project, different options could be explored, such as using more sophisticated error-corrections models (current models are based on the use of probabilistic finite state machines \[35, 36\]) or modifying the generative process that explains the generation of the suffix within an ITP framework. Finally, the latter research direction was explored. Specifically, we provided a refined statistical framework for ITP that outperformed the results obtained by the previous one. Such a statistical framework was applied within an ITP system based on hierarchical models. The details of the proposal can be found in Section 6.2.

Additionally, the ITP system based on stochastic error-corrections models proposed in the first year of the project was initially not integrated in the casmacat workbench, and the reported experiments were not executed on the standard corpora present in the project. The resulting implementation has been used in all the experiments on ITP reported in the second year deliverables. This includes, for instance, the ITP experiments reported in the second field trial of the project (Deliverable 6.2), or some of the experiments of ITP with online learning techniques (Deliverable 4.2).

3.3 Prediction as a Machine Learning Problem

By viewing prediction of sentence completion as a machine learning problem, we develop a classifier that is trained on human post-editing data. The challenge is to extract and test important features for increasing the accuracy of the prediction. As baseline we use the prediction algorithm proposed by \[18\], which uses string edit distance as primary objective, and path score in the search graph as secondary objective. The first step was to determine the accuracy of the algorithm using casmacat’s first field trial data (Section 3.3.1). After the evaluation, candidate features were extracted and tested using Support Vector Machines (Section 3.3.4). Finally, an extended algorithm that included those features was evaluated (Section 3.3.5) using the same dataset as in Section 3.3.1.

3.3.1 Evaluation and Baseline

Before starting with the feature extraction, a first evaluation of the performance of the algorithm \[18\] needed to be made, in order to have a baseline. For this evaluation, we used the first field trial data of casmacat .

The newspaper corpus used is taken from WMT12 \[15\]. It contains articles from CNN, Washington Post, Los Angeles Times, New York Times, Fox News, and The Economist. Five participants were asked to translate the above mentioned corpus from English (source language) to Spanish (target language). The final dataset consists of 1144 post edited sentences; sentences that were translated from scratch were not taken into consideration.

Advanced Post Editing (PIA) using the prediction tool was not tested in the first field trial. It has to be kept in mind that the lack of interactive editing data may falsely decrease the total accuracy of the prediction tool. In the case of interactive editing, the translators may have had accepted an equally correct completion that would have been generated by the search graph. As a result, this evaluation only gives the floor accuracy.

The steps of the evaluation are the following:

---

\[3\] Seventh Workshop on Statistical Machine Translation in Quebec, Canada
For each of the 1144 post edited sentences, it was tested how often the predicted word matched the users’ desired input (the one they had typed in the field trial). The initial match was against the MT output, and for every word there was a mismatch, a new prediction was generated, and the user’s post edited output was then matched against the new translation prediction. These steps are meant to simulate the user’s interactive translation process.

The final accuracy score per sentence was computed by the words predicted correctly, against the not correctly predicted words. The final accuracy is 55.55%.

3.3.2 Absolute Maximum Accuracy of Word Predictions

In order to ensure that the search graphs contain indeed the post edited words that were chosen by the translators, it was measured how often the word that needed to be predicted (i.e. the word that the translator was going to type next) was included in the search graph.

For this reason the same dataset was used, this time including translations that were made from scratch, and the percentage of the existence of the desired word in the search graph was tested. In a total of 61756 Word Predictions (WPs, meaning the completion prediction of the user input; one post edited sentence may have multiple WPs), the percentage is 91.97% (67145 out of 61756 WPs). When we tested only the post edited output, the percentage went up to 93.57% (26343 out of 28153 WPs).

3.3.3 Machine Learning

In order to create a dataset, the first field trial data of casmacat was once again used.

For each sentence, the prediction algorithm was used to output all the (best) matches to the prefix. So, not only the ”winning” prediction (in this case, the one with the lowest string edit distance and highest path score), but all the alternatives. At the same time, candidate features that could be used to enhance prediction were included in the output.

The features are:
- the path score of the matched prefix (user input) and the prediction
- the number of states
- the average path score (score/states)
- the number of deletions (del)
- the number of insertions (ins) and
- the number of substitutions (msm) needed to match the prefix
- their count (sed, total number of edits needed)
- the count averaged by number of tokens of the matched prefix (AvgSed)
- whether the last token of the user input was matched to the last token of the matched string (lastMatched)
- whether the last 2 tokens were matched (last2Matched)
- whether the last 3 tokens were matched (last3Matched)
- levensthein (leven) distance between the last token of the prefix and the matched string (in case it is the same word but in e.g. plural form)
- the prefix size
- whether the user input was larger than the matched string

We restrict ourselves to the 100 best predictions per WP. We wanted to see how often there was at least one accurate prediction in the 100-set of the alternative matches. The final percentage ("oracle") is 71.15% (8937 / 12560 unique WPs), and it indicates the maximum accuracy possible for the prediction algorithm.

The oracle prediction of the full dataset (not only of the top 100 per WP) was also tested. In this case, the percentage of at least one correct prediction in every set of predictions was 79.62% (10001 / 12560 unique WPs)
Table 1: Some of the features tested, and their classification using SVM. From the table it derives that the string edit distance (see), the matching of the last word of the prefix (lastMatched), and the count of substitutions (msm) and insertions (ins) lead to more correct classification, whereas the levenshtein distance (leven) and the number of deletions do not add to the model.

### 3.3.4 Exploring features using SVM

In order to test the candidate features, we used a Support Vector Machine classifier. The dataset was a subset of the 100 best predictions per sentence that contained only 2 lines for each WP; an accurate prediction, and an erroneous one. From this dataset, the 28.93% of the WPs that contained no accurate prediction were excluded, because they would increase the number of false negative classifications.

The dataset was split into training (6586 WPs), development (825 WPs) and test (807 WPs) set. The results of the classification of the test set can be seen on Table 1.

### 3.3.5 Evaluation of the Extended Model

In this part of the task, the prediction algorithm was extended by two of the features separately (lastMatched and sum of edits, sed) and it was evaluated against the first year trial data. As mentioned in section 1.1, the accuracy of the original algorithm among the 1144 post edited sentences of the dataset is 55.55%. When the extended prediction, using the features lastMatched and sed, was tested using same dataset, the accuracy dropped to 55.28% and 55.12% respectively.

### 3.3.6 Conclusion

From the oracle prediction and the fact that 91.97% of the post edited words existed in the search graph, it derives that there is indeed space for improvement in the accuracy of the prediction. However, the features used in this case did not contribute to the overall increase of the completion’s accuracy.

### 4 Task 2: Multi-modality in interactive translation prediction

#### 4.1 Introduction

Typically, post-editing MT output and 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 so that 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, perhaps more natural way of interacting with an ITP or post-editing system is by means of a touch screen or an electronic pen (e-pen). Handwritten words, characters and/or other editing gestures can be naturally and efficiently entered using an adequately designed e-pen interface. Although e-pen interaction may sound impractical for texts that need a large amount of corrections, there are a number of situations where it can actually be more comfortable and effective. First, it can be well suited for post-editing sentences with few errors, as it is the case in translation memory for sentences with high fuzzy matches, or the revision of human post-edited sentences. In these cases, it is worth reminding that e-pen is a just a complementary interface. The traditional keyboard and mouse are always available and the user can keep using at each moment the most comfortable interface, according to her preferences and/or the actual task being done. Second, it would allow to perform such tasks while commuting, travelling or sitting comfortably on the couch in the living room.

A main challenge which emerge when tackling the use of e-pen when post-editing MT and ITP is how to deal with the fact that on-line Hand-written Text Recognition (HTR) systems do commit errors. As the user may weary of using a faulty e-pen system, HTR robustness needs to be improved. This is possible by taking advantage of information derived from the ITP process so as to achieve a system-user synergy which ultimately boosts both e-pen accuracy and usability.

On the other hand, e-pen interaction offers the opportunity to go beyond on-line HTR by allowing the use of specific corrective gestures to improve user efficiency.

4.2 Fundamentals of multi-modal interactive text editing

The development of the CASMACAT e-pen interface follows closely the fundamental principles about multi-modal interaction established in [33]. As in ITP, let $x$ be the input signal (source text in MT), $p'$ a user-validated output text prefix and $s'$ a system-suggested suffix (from a previous ITP step). Now let $t$ be the user feedback in the form of e-pen strokes. This feedback is aimed at accepting or amending parts of $s'$ and/or at adding more text. Now the system has to suggest a new most probable suffix, $\hat{s}$, as a continuation of the prefix $p'$, and conditioned by the on-line pen strokes $t$; that is: $\hat{s} = \arg\max_s P(s \mid x, p', s', t)$.

Here the decoding, $\hat{d}$, of the e-pen feedback, $t$, is “latent” or “hidden”, but using a (“Viterbi-like”) mode approximation, it can be easily uncovered, leading to:

$$\hat{s}, \hat{d} \approx \arg\max_{s,d} P(d \mid x, p', s', t) P(s \mid x, p', s', t)$$

(14)

Taking advantage of the interactive processing context, this optimization problem can be approximately solved in two steps:

1. First solve for $\hat{d}$, using the previous prefix and suffix, and the user feedback (on-line pen strokes), $t$:

$$\hat{d} \approx \arg\max_d P(d \mid x, p', s', t) = \arg\max_d P(t \mid d) P(d \mid x, p', s')$$

(15)

2. Using $\hat{d}$ and $p's'$ (perhaps with the help of some additional e-pen and/or keystrokes to fix possible decoding errors), a new consolidated prefix, $p$, is obtained. Then,

$$\hat{s} \approx \arg\max_s P(s \mid x, p)$$

(16)

which is a conventional ITP equation, where the system relies on the input data, $x$, and a consolidated prefix, $p$, to search for an optimal transcription suffix.
Eq. 15 (right) is similar to the conventional statistical decoding equation for on-line HTR. The first term, \( P(t \mid d) \), is the *morphological likelihood*, which can be modelled, e.g., by character HMM models corresponding to the word(s) in \( d \). The second term is a conditioned prior which can be provided by a special type of *language model*, constrained by information derived from the interaction process; namely: the input data, \( x \), and the previous prefix and suffix, \( p', s' \). Clearly, if these three dependencies are ignored, this term becomes just \( P(d) \), the classical *language model*, typically implemented by means of \( n \)-grams.

As previously commented, the challenge is to actually include these dependencies in the fully conditioned prior \( P(d \mid x, p', s') \) in order to boost the accuracy and robustness of classical on-line HTR systems.

### 4.3 Interactive text transcription using e-pen feedback

Basic concepts and approaches to handwritten text recognition (HTR) and about interactive HTR possibly using e-pen feedback appear in [33] and [29]. These ideas have been developed in several scenarios where the basic task is one of transcription of speech signals [28] and (historic) handwritten document images [32]. These initial developments aimed at word-level interaction; i.e., the tokens to be corrected by means of e-pen strokes are single, full words. But more recent studies aim at developing more flexible interfaces where both character-level and multi-word (including incomplete words) interactive e-pen corrections are allowed. One recent publication of results of these studies can be seen in: Attachment A [22].

### 4.4 Robust on-line HTR by leveraging MT models

The above discussed developments deal with interactive editing the output of (text images) transcription systems. In these works, e-pen accuracy is boosted by taking into account information derived from the interactive transcription process, such as the validated text prefix, the task vocabulary, the input image, etc. Similar ideas can be adopted when the task is ITP (or post-editing). In [3] (see Attachment B) we leverage information from the translation problem to adapt the HTR language model to a specific source sentence. In particular, we take advantage of the source sentence that is being translated, the portion of the translated sentence that has been supervised by the human, and the translation error to be amended. Empirical experimentation suggests that this is a valuable information to improve the robustness of the on-line HTR system achieving remarkable results.

### 4.5 Study of e-pen gestures for ITP

E-pen gestures perfectly complement on-line HTR since together they represent the digital analogy to a proof-reading process. Although there is already a ‘de facto’ standard for gestures for proof-reading the results with state-of-the-art gesture recognizers present high error rates. In [20] (see Attachment C) we present a straightforward solution to incorporate text-editing gestures. Our approach provides disambiguation from handwritten text, excellent accuracy, and an algorithm that is trivial to implement and runs efficiently in any device.

### 4.6 Conclusions

We have shown several works aiming at enabling an MT–e-pen user interface. On the one hand, we have been able to define a set of gestures that cover an important set of editing operations. The recognizer is efficient and accurate. Furthermore, we have improved the robustness of the on-line HTR recognizer by leveraging contextual information. Finally, we have studied how to allow fine-grain editing operations by allowing character level and multi-word editing. To summarize, our work establishes sound bases for a feasible and complete design of an e-pen enabled MT system.
5 Task 3: Prediction from Active Interaction

5.1 Introduction

Traditionally, ITP systems assume the user to systematically supervise each successive translation generated by the system, iteratively finding the point where the next translation error appears until the translation is correct. From the system’s point of view this is a passive protocol since the system just waits for the human feedback, without concern about how the supervision decisions are made. We study the use of active interaction protocols where the system informs the user about which translation elements are worthwhile to supervise. Since not all translations are supervised, “perfect” translations cannot be assured but a better trade-off between translation quality and user supervision effort is potentially achievable.

5.2 Development

The proposed active interaction protocol is based on the use of confidence measures (CM) to locate translation errors in which user attention should be focused. At each interaction, the ITP system provides a new translation completion along with information about the reliability of each translated word. Then, those target words considered to be “incorrect”, i.e. unreliably translated words, are highlighted in a different color so the user can identify them easily.

Confidence estimation is usually addressed as classification problem. Given a target language sentence (and potentially other additional sources of information), a set of features is extracted for each translated word. Note that these features can be consider to be individual estimators for the reliability of each word. Then, a model is trained and employed to compute from these features the “probability” for each word of being incorrect. However, in our case CM must be provided within an interactive environment where the user experience is crucial. Therefore, the ability to compute CM in real-time is a key characteristic to be taken into account.

A consequence of the time constraints inherent to interactive environments is that they specifically limit the complexity of the possible classification models. The temporal complexity of a CM is given by the complexity of computing the features plus the complexity of the classification model. The computation of the features usually involves a constant, or at most linear, time complexity given the input string. In contrast, the complexity of computing the quality score, except for the simplest classification models, usually involves more complex calculations that account for most of the complexity of the CM computation.

Taking these considerations into account, we decide to discard the use of a classification model and focus on computing a single feature as a direct estimator of the quality of the words [13]. Given the quality estimator, we can then classify each word as “correct” or “incorrect” depending on whether its quality score excess or not a certain word classification threshold $\tau_w$. Note that other CM could surely provide better performance but their higher computational complexity forbids their use in an interactive environment.

Note that by varying the value of the classification threshold $\tau_w$ we can modify the behavior of the proposed active interaction protocol. Particularly, we can range between a fully-automatic SMT approach where all words are considered as correctly translated ($\tau_w = 0.0$), and a conventional IMT approach where all words are considered as incorrectly translated, namely suitable to be supervised, ($\tau_w = 1.0$).

Regarding the chosen CM, we implement a word-to-word lexicon feature. Formally, given a target language sentence $y_i^j \equiv y_1 \ldots y_i \ldots y_J$ translation of a source language sentence $x_i^j \equiv x_1 \ldots x_j \ldots x_J$, we follow [34] estimating the reliability, $\Upsilon(y_i, x_i^j)$, of each target word $y_i$
given the source sentence $x^j_1$ as the maximal lexicon probability of the contribution of the word to the total probability of translation $y^i_1$ according to a Model-1 [7]:

$$\Upsilon(y^i_1, x^j_1) = \max_{0 \leq j \leq J} P(y^i_1 | x^j_1) \quad (17)$$

where $x_0$ is the “empty” or “null” word, introduced to capture a target word that corresponds to no actual source word, and $P(y^i_1 | x^j_1)$ is the word-to-word lexicon, namely the probability of target word $y^i_1$ of being the translation of source word $x^j_1$.

We choose this particular CM because it relies only on the source sentence and the proposed translation, and not on an $N$-best list of translations or an additional estimation layer as many other features do [6, 30]. Thus, it can be calculated very fast during search, which, as we have said above, is crucial given the time constraints inherent to interactive systems. Moreover, its accuracy ($\sim 80\%$ precision with $\sim 20\%$ recall) is similar to that of other features as the results presented in [6, 30] show.

### 5.3 Results

We carried out an initial set of experiments involving an “in-laboratory” study [14] to analyze the proposed active interaction protocol. We first measured the accuracy, as measured in classification error rate (CER), of the proposed CM. Results in Figure 2 show that the proposed CM was able to classify the words with less errors than considering all words erroneous ($\tau_w = 0.0$) as conventional ITP systems do, or than considering all words correct ($\tau_w = 1.0$) as fully-automatic SMT systems do. We thus conclude that the proposed active interaction protocol provides useful information that may aid the user in localizing translation errors. Then, we
Figure 4: User interface of the CASMACAT workbench using the proposed active interaction protocol. Words highlighted in red have a large probability of being incorrect translations. Words highlighted in orange are dubious.

studied the potential influence of providing a user with such confidence information. To do that, we defined a simulated user that uses the reference translations of the corpus to automatically test our proposal. At each ITP iteration, the simulated user corrects the next word classified as incorrect by the CM. The results of this experiment, see Figure [3], showed that large reductions in user effort, as measured in Word Stroke Ratio (WSR), can be achieved (∼40% respect to a conventional ITP system) while generating almost error-free translations (∼90% BLEU and ∼5% TER).

A second set of experiments was concerned about how to effectively integrate the proposed active interaction protocol within the actual CASMACAT workbench. In this experiments, we informed users about the reliability of translated words under two different criteria. On the one hand, we highlighted in red those translated words that were likely to be incorrect. We used a threshold that maximizes precision in detecting incorrect words. On the other hand, we highlighted in orange those translated words that were dubious for the system. In this case, we used a threshold that maximizes recall. A description of how these thresholds are computed can be found in [11]. Figure [4] shows a example translations shown to the users working with the CASMACAT workbench using the proposed active interaction protocol.

The objective of this experiment was to measure to which extent actual human users can benefit from the confidence information provided, and which are the factors that maximize this benefit. Deliverable D1.2 provides a description of the users that took part in this pilot experiment.

Prior to any comparison question between the conventional ITP approach and the proposed active interaction protocol, we asked the users a few questions to evaluate to which extent active interaction have matched their expectations. Clearly, these are qualitative aspects that cannot be expressed by a quantity or a measured value. Thus, the users were asked to provide a yes / no response to each question. We also allow the users to clarify their answers whenever necessary. The users’ responses were as follows:

<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you consider active interaction to be a desirable feature?</td>
<td>40%</td>
<td>60%</td>
</tr>
<tr>
<td>Do you consider the provided confidence information to be accurate?</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>Do you consider the active interaction protocol to be annoying?</td>
<td>80%</td>
<td>20%</td>
</tr>
</tbody>
</table>

As appears from the answers and the corresponding clarifications of the users, active interaction is a quite well-regarded feature that an important percentage of them would like to have in a potential translation workbench. However, the users were quite disappointed by the apparently poor performance of the CM. This perception was what made the system annoying for the users. As stated by one participant:
“Many times the words marked by the system as wrong were actually correct, while wrong translations remained in black. In the end I had to double-check most of the sentences to make sure that words marked in black were actually acceptable translations.”

This was a quite surprising result given the good performance previously reported for the chosen quality estimator in laboratory experiments \[6, 30\]. The clarifications made by the users revealed that the main problem stems in the tendency of the system to classify as incorrect words that from the user point of view are clearly correct. For example, proper names are usually classified as incorrect since they tend to appear few times, if any, in the training data. Such errors are infrequent, so they do not penalize much the performance of the estimator as measured in CER or other automatic measures. However, these errors are quite annoying for the users who then distrust the provided CM.

Regarding the comparison between active interaction (AI) and the conventional ITP interaction, it was mainly referred to usability aspects such as the potential difference in translation productivity between the two approaches. Again, these are qualitative aspects for which a yes / no answer, plus a possible clarification, was asked. Next, we provide a summary of the responses given by the users:

<table>
<thead>
<tr>
<th></th>
<th>ITP</th>
<th>AI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which approach do you consider to be more user-friendly?</td>
<td>40%</td>
<td>60%</td>
</tr>
<tr>
<td>Which approach do you consider to be more useful?</td>
<td>60%</td>
<td>40%</td>
</tr>
<tr>
<td>Which approach do you consider to be more productive?</td>
<td>60%</td>
<td>40%</td>
</tr>
</tbody>
</table>

From the answers of the users we can infer that they considered active interaction as an interesting workbench feature that has the potential to improve the usability of the conventional ITP approach. However, some users did not consider that the active interaction protocol could improve the usefulness nor the productivity of ITP systems. Again, the reason for this apparent mismatch in the users’ opinions stemmed in the poor accuracy perceived of the confidence estimations provided. Nevertheless, the users reckoned that active interaction has a great potential to be explored and that it would be a much desirable characteristic whenever appropriate quality information is provided. Quoting one of the participants:

“I could definitely benefit from this type of visual aid, but the system still needs to make better predictions.”

Finally, we would like to discuss the opinions of the users regarding the chosen way to display confidence information within the workbench. As we have described before, we highlight some of the translated words according to two different criteria: words with a high probability of being incorrectly translated and almost surely must be corrected by the user (red color), and dubious words that should be checked by the user but do not necessary have to be wrong (orange color). The users agreed that the specific selection of colors was adequate allowing for an easy identification of the different word types. However, their opinions were mixed regarding the usefulness of the different criteria. Some considered that identifying incorrectly translated words has to be the priority, some others considered that detecting dubious parts of the translation has more interest, and other users even consider that both criteria are equally useful, and thus, both of them should be displayed. As a consensus, we conclude that both criteria must be computed but it should be up to the user to decide which of them, or both, to use to highlight the words.
5.4 Conclusions

Empirical results have shown that users regard active interaction as a desirable feature that should be integrated in a potential translation workbench. However, the CM used in the current implementation was perceived as too error-prone which annoys the users and penalized the usability of the system. A common complaint of the users was that the system systematically classifies as incorrect proper nouns, which is reasonable given that these names are usually out-of-vocabulary words. Nevertheless, users reckoned active interaction as a promising approach if a more reliable confidence information were to be deployed. In the remaining time allotted to this task, we plan to explore how to achieve more reliable confidence information. One direction we intend to explore is to use a proper noun recognizer with the purpose of classifying such words directly as correct translations.

6 Task 4: Prediction from Parse Forest

6.1 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.

The work performed during the last period of activity of this task focused on two approaches to develop grammar-based ITP systems: (1) exact search for partial translations and (2) approximate search for user prefixes. The first effort was 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. Even though the grammar-based ITP systems developed are fully functional, evaluating such schemes goes beyond the scope of casmacat, and thus grammar-based models will not be evaluated in the field trials.

6.2 ITP using Hierarchical translation Models

6.2.1 Introduction

The conventional ITP approach described in [4] has, in our opinion, two basic problems for which we think there is room for improvement. The first problem arises when the system cannot generate the prefix validated by the user. To solve this problem, the authors simply provide an ad-hoc heuristic error-correction technique. The second problem is how the system deals with word reordering. Particularly, the models used by the system were either monotonic by nature or non-monotonic but heuristically defined (not estimated from training data).
6.2.2 Development

In [12], see Attachment D, we work on the foundations of Barrachina et al., [4] and provide formal solutions to the two challenges of conventional ITP approaches. On the one hand, we adopt the statistical formalization of the ITP framework described in [27], which includes a stochastic error-correction model in its formalization to address prefix coverage problems. Moreover, we refine this formalization proposing an alternative error-correction formalization for the ITP framework. Additionally, we also propose a specific error-correction model based on a statistical interpretation of the Levenshtein distance [21]. These formalizations provide a unified statistical framework for the ITP model in comparison to the ad-hoc heuristic error-correction methods previously used.

In order to address the problem of properly deal with reordering in ITP, we introduce the use of hierarchical MT models [10, 38]. These methods provide a natural approach to handle long range dependencies and allow the incorporation of reordering information into a consistent statistical framework. In [12], we also describe how state-of-the-art hierarchical MT models can be extended to handle ITP. Moreover, since the proposed ITP approach is based on the use of hypergraphs, and word-graphs constitute a particular case of hypergraphs, we are able to manage both phrase-based and hierarchical translation models in a unified ITP framework.

6.2.3 Conclusions

We evaluate the proposed ITP approach on two different translation task. The comparative results against the IMT approach described by Barrachina et al., [4] and a conventional post-edition approach show that our ITP formalization for hierarchical SMT models indeed outperform other approaches. Moreover, laboratory experiments indicate that large reductions in the human effort required to generate error-free translations can be achieved.

6.3 Approximate Search for User Prefix

6.3.1 Introduction

We propose two exact dynamic programming algorithms to solve the approximate prefix matching problem for search forests generated by syntax-based translation models. These algorithms find the derivation in the parse forest that have minimal string edit distance (hence "approximate search") and as a secondary criterion maximum derivation score.

We also propose three refinements and assess the impact of the methods on processing speed on the CASMACAT field trial data.

Details are presented in Attachment E (the paper is not yet published).

6.3.2 Summary of Methods

The methods build on the parse forest that is generated during the search for the best translation for the input sentence. It is a compact data structure that encodes a multitude of derivation trees.

Given the parse forest generated by the SCFG search and the user prefix (the partial translation already entered by the user), the interactive machine translation problem is to find the most probably completion of the sentence based on a match of the prefix against the parse forest with the minimal number of edits.
Top-down search algorithm — The first algorithm we propose solves the problem of matching prefix against parse forest with top-down dynamic programming search. The motivation for top-down search is that the user input has to be matched only partially, and parts of the search forest that cover the rest of the sentence do not have to be considered in the matching process. By traversing the parse tree top down, matching the user prefix left-to-right, we can restrict the search to the relevant part of the parse forest. For each edge and node we cache the information about how it can matched, given a start position in the prefix and the maximum number of allowed edits.

Bottom-up search algorithm — Our second algorithm matches the user prefix against the parse forest bottom-up. The main idea is that a leaf edge typically matches the prefix only in one position, so only that information needs to stored. Moving up the tree, we can compute for each edge and node a set of matches against the prefix, supported by an efficient representation of the matches.

Reduction of spurious ambiguity in parse forest — Most statistical machine translation system allow for multiple derivations that yield the same surface string translations. This is also the case in SCFG grammar models, especially hierarchical phrase-based models. Such spurious ambiguity can be detected straightforwardly. Removing irrelevant edges from the parse forest reduces processing time for our algorithms and can be done offline.

Normalizing of non-matched words — Words in the parse forest that do not match any of the words in the user prefix can be normalized, i.e., all replaced with a junk token, since they all will have to treated as insertions or substitutions during matching. After doing so, we can carry out another round of spurious ambiguity reduction.

Inside-outside pruning of parse forest — For each edge and node in the parse forest, we can compute the score of the best derivation that contains it. To reduce the number of edges and nodes, we may discard the ones that can only be part of bad derivations. This is risky, since we may discard better matches (from the view of string edit distance to the user prefix), but has parallels to pruning settings during decoding, and may be required to enable efficient response time.

6.3.3 Conclusions

Of the two algorithms, the top-down search outperforms the bottom-up search, it is about 3–4 times faster in our implementation with similar asymptotic behavior.

We proposed three refinements. Our experiments show risk-free gains for removal of spurious ambiguity, but not for a more aggressive variant that normalizes all words in the parse forest that do not match any word in the user prefix. The third refinement is effective in trading off processing speed against match accuracy.

We plan to further validate this work with syntax-based models that use linguistic constituency labels, carry out experiments for other language pairs, especially German–English, where such syntax-based models have shown superior translation quality, and integrate the method into the CASMACAT workbench.
References


Attachment A

Task 2.3

Daniel Martín-Albo, Verónica Romero, and Enrique Vidal.

Interactive off-line handwritten text transcription using on-line handwritten text as feedback.

Interactive Off-line Handwritten Text Transcription
Using On-line Handwritten Text as Feedback

Daniel Martín-Albo, Verónica Romero and Enrique Vidal
Universitat Politècnica de València, ITI
{dmartinalbo, vromero, evidal}@dsic.upv.es

Abstract—Handwritten Text Recognition has gained attention in the last years mainly due to the interest in the transcription of historical documents. However, automatic transcription is ineffectual in unconstrained handwritten documents, thus human intervention is typically needed to correct the results, even though post-editing is generally inefficient and uncomfortable. To alleviate these problems, multimodal interactive approaches have begun to emerge in the last years. In this scheme, the user interacts with the system by means of an e-pen. This multimodal feedback not only allows us to improve the accuracy of the system but also increases user acceptability. In this work, we present a new approach for interaction based on character sequences. We present developments that allow taking advantage of interaction-derived context to significantly improve feedback decoding accuracy. Empirical tests suggest that, despite the loss of the deterministic accuracy of traditional peripherals, this approach can save significant amounts of user effort with respect to non-interactive post-editing correction.

I. INTRODUCTION

In general, Handwritten Text Recognition (HTR) is difficult because of the inherently variable and noisy nature of the objects (character and word images) to be recognized, among many other adversities. Therefore, as in many other Pattern Recognition problems, recognition results are not (and probably will never be) directly usable in many applications. To cope with this situation, approaches have begun to emerge in the last years in which the user and the system interact hand-in-hand to improve the accuracy of the system. For example, Civera et al. [1] proposed an interactive approach to language translation and Vidal et al. [2] applied a similar interactive approach to speech transcription.

Following similar ideas, Toselli et al. [3] proposed an interactive approach to transcription of handwritten text called Computer Assisted Transcription of Text Images (CATTI). Experiments have proven that this kind of systems can save significant amounts of overall human effort. In [3], an interactive handwritten transcription (IHT) system was proposed where user interaction was always performed at whole word level; that is, the user must detect and correct complete word errors. Continuing the previous work, Romero et al. [4] introduced a new form of interaction based only in character-level corrections.

Furthering the goal of making the interaction with the system more comfortable to the user, Toselli et al. [5] presented a multimodal HIT system using e-pen corrections at whole-word level. The underlying idea behind this work is that the use of more ergonomic interfaces should result in an easier and more comfortable computer-human interaction. However, this kind of feedback leads to the loss of determinism in the interaction feedback. That is, if we use a deterministic interface as the keyboard, the system knows what we have typed with 100% accuracy. The use of a non-deterministic peripheral, such as an e-pen, implies that the possible feedback decoding errors may increase the overall interaction cost. Nevertheless, by using contextual information derived from the interaction, the feedback decoding accuracy can be significantly improved over that of an out-of-the-box recognizer, which can not take advantage of the interaction context. Following this multimodal approach, we presented presented a work which focused on e-pen interactions at character-level [6]. The results of both works suggest that multimodal interaction, despite the loss of the deterministic accuracy of traditional peripherals, can save significant amounts of user effort with respect to fully manual transcription as well as to non-interactive post-editing correction. However, despite the good results, these works fail to provide a sufficiently comfortable approach since they restrict user corrections to whole words or isolated characters. The ideal system would be one that allows the user to correct any subset of characters of the first mistranscribed word.

The aim of this work is to present a first approximation to the ideal system discussed above. Here we present a new approach on interaction using pen-strokes based on character sequences. In this approach, the user can produce corrections, formed by pen-strokes, to fix a set of characters of the first incorrect word in the transcription. These pen-strokes interactions may represent an isolated character, a word substring or an entire word. However, this new, hopefully more ergonomic and friendlier, interaction level entails important difficulties and challenges which have been addressed in this work. In particular, to cope with word substring feedback, different character language models have been studied. In addition, we have also studied how to take advantage of the contextual information derived from the interaction process to improve the accuracy of the feedback decoding system.

To validate this approach we present a comprehensive set of experiments. Clearly, in these interactive scenarios, assessing system performance should require human work. However, the difficulty and high cost of these tests make more feasible to use the classic pattern recognition assessment paradigm based on labelled corpora to obtain adequate estimates of human effort required to achieve the goals of the considered tasks. In fact, evaluating this interactive system requires on-line data, thus we properly simulated user interactions using an on-line handwritten corpus as in [5]. In this way, the experiments in this paper are fully reproducible.
An interactive framework for handwritten text transcription.

II. INTERACTIVE OFF-LINE HANDWRITTEN TEXT TRANSCRIPTION

As explained in [3], in the CATTI framework the human transcriber is involved in the transcription process. Formally speaking, let \( x \) be a feature vector sequence extracted from a handwritten text line. The IHT system starts proposing a transcription of \( x \). Then, the user validates an initial part of this transcription (\( p' \)), which is error-free, and introduces a correct word (\( u \)) thereby producing a correct transcription prefix (\( p = p'u \)). After that, the system must take into account this information to suggest a new suitable continuation suffix (\( s \)). This process continues until the user accepts the transcription as correct. At each step of this interaction process, the system must take into account the image representation (\( x \)) and the prefix (\( p \)) to search the most likely suffix (\( s \)):

\[
\hat{s} = \arg \max_s P(s \mid x, p) = \arg \max_s P(x \mid p, s) \cdot P(s \mid p)
\]

Since the concatenation of \( p \) and \( s \) constitutes a full transcription hypothesis, \( P(x \mid p, s) \) can be approximated by concatenated character Hidden Markov Models (HMMs) [7], [8]. On the other hand, \( P(s \mid p) \) is approximated by a dynamically modified \( n \)-gram to cope with the increasingly consolidated prefixes [3].

III. USING ON-LINE HANDWRITTEN TEXT AS FEEDBACK

The idea of using on-line handwritten text as feedback has pros and cons. The use of more ergonomic interfaces should result in an easier and more comfortable computer-human interaction. However, this implies the introduction of an on-line HTR decoding system in order to deal with this nondeterministic information.

Continuing with the above notation, let \( t \) be the representation of the input image. Let \( \hat{d} \) be the on-line pen-strokes provided by the user, they are intended to correct errors in the previously suggested suffix (\( s' \)). The position of these corrections allows us to define the user-validated-error-free prefix (\( p' \)). This prefix \( p' \) consists of two parts: whole words (\( p'' \)) and the last incomplete word (\( u' \)). Finally, the system has to find a new suffix (\( \hat{s} \)) as a valid continuation of the prefix (\( p' \)), considering all possible decodings (\( d \)) of the on-line data (\( t \)). An example of this notation is shown in Fig. 1. Following [5], \( \hat{s} \) can be formulated as:

\[
\hat{s} \approx \arg \max_s \max_P(t \mid d) \cdot P(d \mid p', u', s') \cdot P(x \mid s, p', d) \cdot P(s \mid p', d)
\]

Eq. (2) can be approximately solved using a two-step approach as seen in [5]. In the first step, the on-line subsystem must decode the on-line data (\( t \)) into the most probable sequence of characters (\( d \)), knowing that this decoding must be a valid continuation of the prefix (\( p' \)). Once \( d \) is available, a new consolidated prefix \( p \) is produced joining the previous prefix (\( p' \)) and the most probable decoding (\( d \)). Then, the second step searches, in a similar fashion to (1), for the most probable suffix using the new consolidated prefix. These two steps are repeated until \( p \) is accepted by the user as a full correct transcription of \( x \). An example of this process is shown in Fig. 2.

In this work we focus on the first step of (2); i.e., the restricted decoding of the user interactions, which can be isolated in a single expression as:

\[
\hat{d} = \arg \max_d P(t \mid d) \cdot P(d \mid p', u', s')
\]

where, \( P(t \mid d) \) is the morphological model provided by HMMs and \( P(d \mid p', u', s') \) can be approached by a language model dynamically constrained by information derived from the interaction process. Different scenarios arise depending on the assumptions and constraints adopted for \( P(d \mid p', u', s') \).

The first and simplest scenario corresponds to a naive approach where any kind of interaction-derived information is ignored; that is, \( P(d \mid p', u', s') \equiv P(d) \). This scenario will be used as baseline result in our experiments.

A more restrictive scenario arises when we regard the portion of word already validated (\( u' \)). In this case the decoding can be more accurate, since we know beforehand that the sequence of pen-strokes to decode must be a valid continuation of the part of word accepted so far. This scenario can be written as \( P(d \mid p', u', s') \equiv P(d \mid u') \).

The last scenario emerges if we add, to the previous one, the set of complete words \( p'' \). In this case, the possible decodings are constrained to be a suitable continuation of the whole prefix accepted so far. This scenario can be written as \( P(d \mid p', u', s') \equiv P(d \mid p', u') \).

IV. DYNAMIC LANGUAGE MODELING

Language model restrictions are implemented on the base of \( n \)-grams. As we mentioned earlier, the baseline scenario given by \( P(d) \) (being \( d = \{c_1, c_2, ..., c_i\} \)) does not take into account any information derived from the interaction. Here, character \( n \)-grams have been used for modelling \( P(d) \).

The next scenario, given by \( P(d \mid u') \), is approached also using a character \( n \)-gram language model, but it is conditioned by the fragment of word (\( u' \)). Following similar discussion as in [5] this can be written as:

\[
P(d \mid u') = \prod_{i=1}^{n-1} p(c_i \mid c_{i-1}^{i}, u'_{k-n+i+1}) \prod_{i=n}^{l} p(c_i \mid c_{i-n+1}^{i-1})
\]
where $u' = \{u'_1, u'_2, \ldots, u'_t\}$. The first term of (4) accounts for the probability of the $n-1$ character of the suffix, whose probability is conditioned by known characters from the validated prefix, and the second one is the usual $n$-gram probability for the rest of the unknown characters.

The scenario defined by $P(d \mid p', u')$ uses the last incomplete word of the prefix ($u'$) and the complete words of the prefix ($p'$). This scenario has been approached in two different ways. The first one uses a character $n$-gram model conditioned by $p' = p''u'$ (where ' ' is a white space). We can restrict this model in a similar manner to (4):

$$P(d \mid p') = \prod_{i=1}^{n-1} p(c_i \mid c_{i-1}^{p''}, u'; c_{i-n+2}) \cdot \prod_{i=n}^{l} p(c_i \mid c_{i-n+1})$$  \hspace{1cm} (5)

where $p = \{p_1, p_2, \ldots, p_l\}$.

The second approach for $P(d \mid p', u')$ is modeled using a special kind of $n$-gram that combines words and characters. We called this $n + m$-grams, where $n$ accounts for the maximum number of complete words in each $n + m$-gram and $m$ is the number of characters after this $n$ words. Given that we want to recognize a sequence of characters ($d$) and part of our prefix ($p'$) is composed by complete words ($p''$), makes sense to use models that combine two levels of representation (words and characters). This two-level-model approach is not new; this idea has been previously used successfully in speech recognition. For example, Bazzi et al. [9] combined lexical entries with sub-lexical units to generate a hybrid language model.

The main advantage of using $n+m$-grams, instead of normal $n$-grams, is that the former model can be more informed, being smaller. For example, if we model the words "sleep furiously" with the first approximation of $P(d \mid p')$, we would need a total of seven 9-grams to represent the string. In contrast, using, for example, an 1-8-grams we would need only two elements; i.e., "sleep furiously" and "furiously".

More theoretically speaking, let $w = \{w_1, w_2, \ldots, w_l\}$ be a sequence of words. To compute the probability of this sequence with a $(n + m)$ language model, $w$ can be considered split into two fragments, the first part represented as a sequence of complete words and the second part as a sequence of characters $w = \{w_1, w_2, \ldots, w_l, c_1, c_2, \ldots, c_s\}$, where length($w_{k+1}, w_l$) is the length in characters from word $w_{k+1}$ to $w_l$. Considering the boundary point $b$ as a hidden variable, we can write:

$$P(w) = \sum_{1 \leq i \leq l} \prod_{i=1}^{b} p(w_i \mid w_{i-n+1}) \cdot \prod_{i=b}^{l} p(c_i \mid c_{i-1}^{b}) \cdot \prod_{i=b}^{s} p(c_i \mid c_{i-n+1}^{b})$$  \hspace{1cm} (6)

where the first term of (6) accounts for the probability of the $b$ complete words, the second one accounts for the probability of the characters influenced by characters and words and the last one is the usual $n$-gram.

$p(d \mid p', u')$ can be approached by adapting this $n + m$-gram language model so as to cope with the consolidated prefix. The previous equation provides a model for the probability of $p(w)$, where $w$ can be seen as the concatenation of $p''u'd$. But now both $p''$ and $u'$ are fixed. Therefore some changes are needed. Following similar discussion to that presented in [5] and considering only the last complete word of the prefix ($p'' \equiv w_p$) we can write $P(d \mid p', u')$ as:

$$P(d \mid u', w_p) = \prod_{i=1}^{n-k-1} p(c_i \mid c_{i-1}, u', w_p) \cdot \prod_{i=n-k}^{l} p(c_i \mid c_{i-n+1})$$  \hspace{1cm} (7)

where the first term of (7) accounts for the probability of the $n - k - 1$ characters of the suffix, whose probability is conditioned by known characters from the validated prefix and the previous complete word, the second one accounts for the probability of the $n - 1$ characters of the suffix, whose probability is conditioned by characters of the last incomplete word from the validated prefix, and the last one is the usual $n$-gram probability for the rest of the unknown characters.

V. EXPERIMENTAL DETAILS

The details of the HTR systems, the corpora and the assessment measures used in the experiments are given below.
A. Off- and On-line HTR Baseline Systems

Besides the main off-line recognition system, an additional subsystem is needed here in order to cope with the e-pen interactions. In this section a general overview of both HTR systems is presented.

The two systems follow the classic scheme of pattern recognition: preprocessing, feature extraction and recognition.

The first two blocks of each system entail different techniques, since they dealing with different kind of information. However, the last one is the same for both subsystems.

Off-line HTR preprocessing is aimed at correcting image degradations and geometry distortions: skew and slant corrections and size normalization. On the other hand, on-line handwriting preprocessing involves, in this case, only two steps: duplicated point removal and noise reduction.

Feature extraction in the off-line case transforms a preprocessed image into a sequence of feature vectors representing grey levels and gradients. The on-line feature extraction module transforms the preprocessed trajectories into a new temporal sequence of seven-dimensional feature vectors.

The recognition process is based on HMMs. Characters are modeled by a continuous density left-to-right HMMs. A Gaussian mixture is used to model the emission of each HMM state. For each system, the number of Gaussians was 8 for the on-line system and 128 for the off-line.

B. Corpora

Three corpora have been used in the experiments. First, we used the IAMDB corpus to evaluate the off-line system. The IAMDB is a publicly accessible corpus composed of 1,539 scanned text pages, handwritten by 657 different writers. The database is provided at different segmentation levels, here we use sentence-segmented images. To better focus on the essential issues of the considered problems, no punctuation marks, diacritics, or different word capitalizations are included in the transcriptions. From 2,324 sentences that forms the corpus, 200 were used as test, leaving the rest as training.

IAMDB consists of hand-copied sentences from the much larger electronic text LOB corpus, which contains about one million running words. This corpus was used here (after removing the test sentences) to train the language models based on n-grams.

Finally, the on-line UNIPEN corpus was employed to simulate the user e-pen interactions. The UNIPEN corpus comes organized in several categories: lower and upper-case letters, digits, symbols, isolated words and full sentences. Here, three categories were used: digits, lowercase letters and symbols. Three arbitrary writers were chosen as test partition and 17 as training data.

C. Assessment Measures

Two kinds of measures have been adopted to assess the systems. On the one hand, the quality of non-interactive transcriptions can be properly assessed with the well known word error rate (WER). It is defined as the minimum number of words that need to be substituted, deleted or inserted to convert a sentence recognized by the system into the corresponding reference transcription, divided by the total number of reference words. On the other hand, the effort needed by a human transcriber to produce correct transcriptions using an HTR system is estimated by the word stroke ratio (WSR), which can be also computed using the reference transcriptions.

After each system hypothesis, the longest common prefix between the hypothesis and the reference is obtained and the first unmatching word from the hypothesis is replaced by the corresponding reference word. This process is iterated until a full match with the reference is achieved.

Therefore, the WSR can be defined as the number of word level user interactions that are necessary to achieve the reference transcription of the text considered, divided by the total number of reference words. This definition makes WER and WSR comparable. Moreover, the relative difference between them gives us a good estimate of the reduction in human effort that can be achieved by using CATTI with respect to using a conventional HTR system followed by human post-editing. This estimated effort reduction will be denoted as EFR.

Finally, the conventional classification error rate (ER) was used to assess the accuracy of the on-line HTR subsystem.

VI. EXPERIMENTAL RESULTS

The aim of the developed experiments were twofold: 1) check which degree of synergy can actually be expected by taking into account information derived from the interactions; and, 2) study how much effort is increased using an e-pen compared to using a deterministic device, such as the keyboard, during the interaction process.

To address these questions we simulated the process of a user performing a transcription using our system. We employed the IAMDB corpus as a document to be transcribed. For each sentence, the system proposes a new potential transcription. If the answer contains any mistakes, some amendments must be introduced. To simulate this user interactions we used the UNIPEN corpus. After correcting the error and having a new consolidated prefix, the system generates a new suffix. This process runs until the transcription is equal to the reference.

<table>
<thead>
<tr>
<th>TABLE I. STATISTICS OF THE TEST SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Fragments</td>
</tr>
<tr>
<td>IAMDB</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

In order to increase the reproducibility and simplify the experiments, we assumed that the rectifications correct from the beginning of the erroneous fragment until the end of the first mistranscribed word; i.e., if the system recognizes the word January, but the reference is janitor, the correction used here is janitor. Moreover, samples used as corrections were generated as follows: we created three samples for every erroneous sequence to correct, one for each test writer. Each of these samples was generated by binding isolated characters from each writer defined as test from the UNIPEN corpus. Continuing with the previous example, if the amendment

\[ \text{Although in this paper we have focused on correcting word segments, we believe that correcting a word segment costs as much as correct the whole word.} \]
needed is the sequence itor, we need to paste together an i, a t, an a and a r of each user. These characters are chosen at random. Table I shows the statistics of the data employed in the experiment.

| TABLE II. FEEDBACK DECODING ERROR RATES |
|-----------------|-----------------|-----------------|-----------------|-----------------|
|                 | CN  | CNp | W-CNp | M-CNp | Comb. |
| Word Fragments  | 11.4| 3.2 | 8.5   | 8.4   | 3.2   |
| Complete Words  | 12.7| 12.7| 10.9  | 8.5   | 8.5   |
| Average         | 12.5| 11.0| 10.5  | 8.5   | 7.5   |

Table II reports the average feedback decoding error considering the different scenarios described before. The first one, called here CN, corresponds to the baseline given by \( P(d) \). Here we use a 9-gram character model constructed using the characters of isolated words since in this scenario there is no need of context between words. The second scenario \( (P(d | u')) \), called here CNp, uses the same character 9-gram language model as above, but in this case is prefix-conditioned. The third one, called W-CNp, is a a whole-prefix-conditioned character 9-gram \( (P(d | p', u')) \). This language model has been constructed using separated characters of words grouped in pairs, thus this language model contains information about how words are connected. The fourth one, named by M-CNp, is a whole-prefix-conditioned 1-8-gram. Finally, the last column represents our best system, created by combining the best results (CNp for word fragments and M-CNp for complete words).

As expected, the more information available, the highest feedback decoding accuracy. The excellent result achieved by CNp recognizing word fragments may be due to the way that language model is created, since it only uses isolated words. Honestly speaking, it is also possible that the use of the prefix beyond the intra-word context when recognizing word fragments can mean more hassle than help.

As a final overview, Table III compares the results at the user effort level. The first column shows the WER achieved using post-editing corrections. The second one, shows the WSR achieved using an IHT system using a deterministic interface, such as the keyboard. The third column shows the WSR for our best feedback decoding approach (last column in Table II). This value is calculated under the simplification that if the system fails to recognize a sequence of characters, the user proceeds to edit it again with the keyboard (thereby combining two corrections). Finally, the last two columns show the overall estimated effort reductions (EFR) for the deterministic IHT system and our multimodal IHT approach with respect to post-edition with auto-completing.

| TABLE III. EFFORT COMPARISON RESULTS |
|-------------------------------|-----------------|-----------------|-----------------|-----------------|
| Post-editing WER              | Word Stroke Ratio | Overall EFR      |                 |                 |
| Keyboard e-pen                | 25.1            | 21.5            | 23.1            | 14.3            | 8.0             |

According to these results, the expected user effort for our approach, is only slightly higher than that of using a deterministic interaction system.

VII. CONCLUSIONS

In this work, we have studied a new way of interaction using pen-strokes as feedback. Here, this feedback is used as a part of a prefix to improve the transcription given by the computer. Thus, the system proposes a new suffix that the user can accept as a final transcription or modify in an iterative way until a full and correct transcription is finally produced.

Empirical tests presented here support the benefits of using this approach rather than traditional HTR followed by human post-editing. From the results, we observe that the use of the more ergonomic feedback modality comes at the cost of only a reasonably small number of additional interaction steps needed to correct the few feedback decoding errors. The number of these extra steps is kept very small thanks to the system ability to use interaction-derived constraints to considerably improve the on-line HTR feedback decoding accuracy.

Finally, as future work we intend to further explore the use of \( n + n \)-grams since its potential is clear.

ACKNOWLEDGMENT

This work was partially supported by the Spanish MEC under the STrAIDA research project (TIN2012-37475-C02-01), the MITTRAL (TIN2009-14633-C03-01) project, the FPU scholarship AP2010-0575, by the Generalitat Valenciana under the grant Prometeo/2009/014, and from the EU’s 7th Framework Programme (FP7/2007-2013) under grant agreement n. 600707 – transcSpeaker and n. 287576 – CasMaCat.

REFERENCES

Attachment B

Task 2.3

Vicent Alabau, Alberto Sanchis, and Francisco Casacuberta.

Improving on-line handwritten recognition in interactive machine translation.

*Pattern Recognition, 2013.*

*In press.*
Improving on-line handwritten recognition in interactive machine translation

Vicent Alabau, Alberto Sanchis, Francisco Casacuberta

PII: S0031-3203(13)00410-X
DOI: http://dx.doi.org/10.1016/j.patcog.2013.09.035
Reference: PR4942

To appear in: Pattern Recognition

Received date: 26 July 2012
Revised date: 11 June 2013
Accepted date: 30 September 2013

Cite this article as: Vicent Alabau, Alberto Sanchis, Francisco Casacuberta, Improving on-line handwritten recognition in interactive machine translation, Pattern Recognition, http://dx.doi.org/10.1016/j.patcog.2013.09.035

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting galley proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
Improving On-line Handwritten Recognition in Interactive Machine Translation

Vicent Alabau*, Alberto Sanchis*, Francisco Casacuberta*

Institut Tecnològic d’Informàtica, Universitat Politècnica de València, Camino de Vera s/n, 46022 València, Spain

Abstract

On-line handwriting text recognition (HTR) could be used as a more natural way of interaction in many interactive applications. However, current HTR technology is far from developing error-free systems and, consequently, its use in many applications is limited. Despite this, there are many scenarios, as in the correction of the errors of fully-automatic systems using HTR in a post-editing step, in which the information from the specific task allows to constrain the search and therefore to improve the HTR accuracy. For example, in machine translation (MT), the on-line HTR system can also be used to correct translation errors. The HTR can take advantage of information from the translation problem such as the source sentence that is translated, the portion of the translated sentence that has been supervised by the human, or the translation error to be amended. Empirical experimentation suggests that this is a valuable information to improve the robustness of the on-line HTR system achieving remarkable results.

Keywords: interactive pattern recognition, on-line handwritten text recognition, interactive machine translation, human interaction

*Corresponding author: Phone: (34) 96 387 70 69, Fax: (34) 96 387 72 39.

Email addresses: valabau@iti.upv.es (Vicent Alabau), asanchis@iti.upv.es (Alberto Sanchis), fcn@iti.upv.es (Francisco Casacuberta)
1. Introduction

Since the breakout of tactile smartphones, the number of devices featuring a touch-screen has been increasing at a fast pace. The success of tactile smartphones has fostered a new kind of keyboardless technology: the tablet computers. They have been presented as a substitute of paper notebooks although the possibilities this new technology may provide are still to be unveiled.

In that context, on-line handwritten text recognition (HTR) can play a crucial role. First, because to input text in such devices using a virtual keyboard is far from the efficiency of regular keyboards. Secondly, handwriting is a natural way to communicate. Withal, an HTR interface can commit recognition errors. Thus, if the HTR system is not robust enough, user experience could be negatively affected hindering its use. In this regard, many works have tried to improve HTR accuracy. Primarily focusing on feature extraction and modeling [1, 2, 3]. Other authors have tackled the problem of automatically correcting errors from the system output in order to provide a more accurate input to higher-level applications. For instance, Quiniou et al. [4] propose a technique to improve the performance of a HTR system by obtaining a consensus hypothesis out of a n-best lists, and then, characterizing the errors and correcting them. Similarly, Farooq et al. [5] use a translation model to conduct an automatic post-editing. Additionally, Devlin et al. [6] used a machine translation system to rerank an OCR n-best list. The idea was that easily translatable options would have a better syntax, which in the end resulted in small accuracy improvements. Nevertheless, those works did not leverage any contextual information of the specific task at hand, a topic that, in our opinion, has received little attention. Following this line of research, Toselli et al. [7] explored the use of on-line HTR for interactive transcription of text images. In that work, the user was expected to correct erroneously recognized words by handwriting the correction using a tactile display. The authors took advantage of the erroneously predicted word and the previous one to improve HTR robustness. Inspired by Toselli et al. [7, 8], we address the problem of using an on-line HTR system to correct the errors in a machine translation (MT) application. However, in this work, we aim to improve HTR performance by exploiting the different knowledge sources which are involved MT applications.

Usually, state-of-the-art MT systems cannot perform translations to fit quality demands by the translation industry. Hence, it is typical to have the automatically produced output documents revised by a professional translator. In this manual process, known as post-editing (PE), the human expert can spend hours of work to achieve high-quality translations. Interactive machine translation (IMT) [9, 10, 11] was developed to deal with this problem. In IMT, a human expert is introduced in the middle of the translation process. This way, she can amend errors from the system output and useful feedback is used by the system to automatically improve the part of the translation to be revised.

The usual way to introduce the corrections in MT and IMT is by means of the keyboard where the mouse is used to fix the position [12]. However, other interaction modalities are also possible. For example, speech interaction was studied in [13, 14, 15]. There, several scenarios were proposed, in which the user was expected to utter aloud parts of the current hypothesis along with one or more corrections. Later, we proposed the use of on-line HTR to IMT in [16, 17]. To our knowledge, our work has been the first approach to on-line HTR in IMT so far. Nonetheless,
those works presented very preliminary results explaining simple contextual models and HTR interaction restricted to
isolated words.

In this paper we present relevant novelties with respect to previous work that can be summarized in two main
improvements. First, we introduce a new HTR model that leverages state-of-the-art phrase-based models, whereas
previous work was based only on word-based translation models. Second, we extend the interaction scheme to allow
sequences of words (phrases) to be written and not just isolated words. In addition, we propose a method to recover
efficiently from HTR errors using contextual menus. Finally, a new and exhaustive experimental study is presented to
evaluate all those novel contributions and preliminary ideas.

The remainder of this paper is organized as follows. First, the process to produce high-quality translations is
introduced in Sec. 2. Second, in Sec. 3 several alternatives to incorporate contextual information from the translation
problem into the HTR decoding will be explored. Section 4 is devoted to the evaluation of the proposed models.
Finally, conclusions and future work will be discussed in Sec. 6.

2. Producing High-Quality Translations

In the last years, machine translation (MT) has become a strategic asset in the translation industry. MT is used
to speed up the translation process since it enables the automatic translation of large amounts of documents. In this
context, MT is approached under a statistical framework, due to the fact that statistical MT allows companies to
build customized, topic-specific MT systems very economically. Here, the problem consists in finding the most likely
translation \( \hat{t} \) in a target language given a source sentence \( s \) in a source language,

\[
\hat{t} = \arg\max \Pr(t \mid s)
\]  

which can be modeled in different ways \cite{18}.

2.1. Post-editing a Machine Translation Output

Although leveraging MT can be very convenient, it is usually the case that the translation quality does not meet
the user requirements. Thus, the MT output must be revised. The process of revising and amending the system output,
known as post-editing (PE), consists in deleting, inserting, substituting and swapping text from the MT output to
achieve the desired quality in the translation. This is an expensive task, since the users should review the whole output
and correct manually the translation errors. In the cases in which the automatically produced translations are of low
quality, PE can eventually require more effort than manually translating the source input from the scratch. Moreover,
in PE, the system does not take advantage of the human corrections.

2.2. Interactive Machine Translation

The MT paradigm is shifting slowly but steady towards an interactive MT scenario (IMT). In IMT \cite{9, 10, 11} the
system goal is not to produce translations in a completely automatic way and then perform a completely unassisted
On the contrary, IMT aims at building the translation collaboratively with the user as a professional advisor, so that the effort to produce a satisfactory output is minimized.

A typical approach to IMT is shown in Fig. 1. A source sentence $s$ is given to the IMT system. First, the system outputs a translation hypothesis $\hat{t}$ in the target language, that would correspond to the output of fully automated MT system (i.e., based on Eq. (1)). Next, the user analyzes the source sentence and the current hypothesis, and validates the longest error-free prefix $p$ finding the first error. Then, the user amends the erroneous word by typing the correct word $d$. Based on this amendment, the system creates a new validated prefix $p \cdot d$, with $\cdot$ as a concatenation operator. With that information, the system is able to produce a new, hopefully improved, translation $\hat{t}$ that is coherent with the information provided, that is, $p \cdot d$ must be a prefix of the new $\hat{t}$. This process is repeated until the user agrees with the quality of the resulting translation. In this work we assume that this protocol is performed left-to-right, but other protocols are also possible.

The iterative nature of the process is emphasized by the loop in Fig. 1, which indicates that, for a source sentence to be translated, several interactions between the user and the system could be performed. In each interaction, the system produces the most probable translation $\hat{t}$ that is coherent with the prefix formed by concatenating the previous prefix $p$ and the user correction $d$:

$$\hat{t} = \arg\max_{t \cdot \tau(t, p \cdot d) \cdot \tau(t, p \cdot d)} \Pr(t \mid p, d, s)$$

where $\tau(t, p \cdot d)$ is a function that is true if $p \cdot d$ is a prefix of $t$. It is worthy of note that the main difference between Eq. (1) and Eq. (2) is that, in the second case, $\hat{t}$ is must be coherent with the validated prefix $p \cdot d$. Since the probabilistic models in Eq. (1) and Eq. (2) are estimated in the same way, Eq. (2) can be considered as a constrained search problem of the classical MT problem. In fact, at the beginning, when the user has not validated any prefix, Eq. (1) and Eq. (2) are equivalent equations. In addition, adaptive approaches can also be assumed, where the system is able to learn from each user interaction to improve the underlying statistical models [19].

For the sake of a better understanding, a typical translation IMT session is exemplified in Fig. 2. First, the system starts with an empty prefix, so it proposes a full hypothesis. Then, the user corrects the first error, not, by typing ‘is’. Next, the system proposes a new suffix, in which the first word, not, has been automatically corrected. The user amends ar by typing ‘in’. Finally, as the new proposed suffix is correct, the process ends. Note that 4 operations would...
have been needed in a PE scenario, whereas only 2 are needed in IMT. In this example, the user types the complete wrong word. Nevertheless, it is straightforward to extend this operation to the character level instead of word level.

**SOURCE** (s): si alguna función no se encuentra disponible en su red

**REFERENCE** (r): if any feature is not available in your network

<table>
<thead>
<tr>
<th>ITER-0</th>
<th>(p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>if any feature not is available on the network</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ITER-1</th>
<th>(p)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>if any feature is not available at the network</td>
<td></td>
</tr>
<tr>
<td>(p)</td>
<td>if any feature is not available</td>
<td></td>
</tr>
<tr>
<td>(d)</td>
<td>if any feature is not available</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ITER-2</th>
<th>(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>if any feature is not available in your network</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FINAL</th>
<th>(i ≡ r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>if any feature is not available in your network</td>
</tr>
</tbody>
</table>

Figure 2: Example of an IMT session for translating a Spanish sentence s to an English sentence t. Initially, in iteration 0, the prefix is empty, i.e., the user has not performed any validation. In iteration 1, the system proposes a fully automatic translation \( \hat{t} \). Then, the user finds the first error and amends it by introducing the correct word \( d \), which is shown in **boldface**. As a result, the user has implicitly validated a prefix \( p \), shown in *italics*. The concatenation of the prefix and the corrected word constitutes a new prefix for the next iteration (displayed underlined). The process continues until the user is satisfied with the solution. Note that 4 operations would have been needed in a PE scenario, whereas only 2 are needed in IMT.

### 3. Using On-Line HTR to Correct MT Output

Typically, the correction of MT output is performed using a keyboard and, occasionally, a mouse to position the cursor [12]. Professional translators agree that this approach has been proved to be efficient. 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. Thus, although e-pen interaction may sound impractical for texts that need a large amounts of corrections, there is a number of circumstances where e-pen interaction can be more suitable. For example, it can be well suited for amending sentences with few errors, as the revision of human post-edited sentences, or translations where the system has a high confidence that the output is of good quality. Furthermore, it would allow to perform such tasks while commuting, traveling or sitting comfortably on the couch in the living room.

Now, imagine an application devised to translate documents. On the one hand, there is a text area with the output of an automatic machine translation system. As this output may contain errors, the user of the application reads the output to locate the first error. The reading is performed in a specific order, left-to-right in most western languages, for
instance. Let us assume that when the user finds the first error, all the words before it have already been revised and validated. Thus, they can be regarded as correct. Once the error has been located, the user introduces the correction with a stylus. As a result, the system receives a position where the error is located, a word that is incorrect (the word pointed by the position) and a sequence of pen strokes that represent the correct word in that position. On the other hand, the source document to be transcribed is shown to the user. There is a strong relationship among the words in the source sentence and the words in the target sentence.

Figure 3 is a mock-up of a possible application on a tablet device for such scenario. The screen is divided in two sections. First, the upper part shows the source document, and probably the source sentence being currently translated, \( s \), is highlighted appropriately. Second, the lower section contains the current state of the translation, \( t \).

Since we assume that post-editing is usually performed from left to right, the text which has already been revised and validated is highlighted. On the other hand, the text which is to be revised is displayed grayed out. From the sentence currently being translated we can identify three parts: the revised prefix of the sentence, \( p \), the error committed by the system, \( e \), and the correction proposed by the user introducing strokes with a stylus, \( x \).

Figure 3: Mock-up of an interactive machine translation application on a tablet device.

In a scenario as described above, the HTR subsystem should make few errors to make the application usable. The aim of this work is to devise a robust HTR system that allows a potential user to revise and correct the output.
of a machine translation system using an electronic pen. To this regard, we assume that the user will introduce the corrections by writing over the word or sequences of words (phrases) she judges to be incorrect. Thus, the problem of on-line HTR consists in converting a sequence of strokes, $x$, into a word or phrase in text format, $d$. The strokes can be acquired from a stylus, electronic pen or a touch-screen.

3.1. System Baseline

The baseline approach to the problem from a statistical point of view is to obtain the most likely decoding $\hat{d}$ given the strokes $x$, 

$$\hat{d} = \arg\max_d Pr(d \mid x) = \arg\max_d Pr(d) Pr(x \mid d)$$

(3)

where $Pr(d)$ can be represented by a language model and $Pr(x \mid d)$ by morphological models.

The morphological models can be modeled by hidden Markov models [2] or neural networks [1]. On the other hand, a common and practical approach to model $Pr(d)$ is by means of $n$-grams [20]. The description of an on-line HTR system would end here for most applications. However, our purpose is to take advantage of the information available in the MT application to make on-line HTR more robust. In the remainder of this section, we will introduce gradually the different kinds of information sources into the language model. With the addition of each of them, we aim to make the on-line HTR system more robust.

3.2. Discarding the produced error

In the e-pen enabled interface aforementioned, the user is expected to write the strokes over the erroneously translated word, and thus, the system knows what word the user wants to replace. Therefore, the first and easiest approach is to remove the erroneous word $e$ from the list of candidate hypotheses. This way, Eq. (3) becomes

$$\hat{d} = \arg\max_{d \neq e} Pr(d) Pr(x \mid d)$$

(4)

3.3. Exploiting information from the revised translation

The second sensible approach to take is to add information regarding the revised translation prefix, $p$. Again, from Eq. (3) we can derive an HTR system that takes into account previously validated words:

$$\hat{d} = \arg\max_d Pr(d \mid x, p) \approx \arg\max_d Pr(d \mid p) Pr(x \mid d)$$

(5)

under the assumption that $Pr(x \mid d, p)$ does not depend on $p$ if $d$ is known. In addition, $Pr(d \mid p)$ is a prefix language model, i.e., the probability of $d$ depends on the left-context. Of course, we can also discard the erroneous word from Eq. (5),

$$\hat{d} \approx \arg\max_{d \neq e} Pr(d \mid p) Pr(x \mid d)$$

(6)

These techniques can be extrapolated to most post-editing tasks. In fact, Toselli et al. [7] used the erroneous word and a 2-gram model to improve the HTR performance for interactive transcription of text images. Next, we will show how the information regarding the translation process can be exploited for further improve HTR decoding.
3.4. Leveraging information from the source sentence

A specific source of information that can help to improve robustness in the MT scenario is, naturally, the sentence in the source language. Since the target sentence conveys the meaning of the source sentence, \( s \), user corrections should be restricted somehow to the possible translations of it. Hence, we can formulate the problem as,

\[
\hat{d} = \arg \max_d Pr(d \mid x, p, s) \approx \arg \max_d Pr(d \mid p, s) Pr(x \mid d)
\]

assuming that \( Pr(x \mid d, p, s) \) does not depend on \( p \) and \( s \) if \( d \) is known.

Nevertheless, the relationship between the target and the source sentence in \( Pr(d \mid p, s) \) is not trivial to establish.

Two possibilities are considered in this work. First, word-based models are the basis for modern statistical MT [21]. Although they cannot provide a good performance when translating complete sentences, they offer a smoothed and reliable probability distribution for word models. In addition, they serve as initialization for the second kind of models considered: phrase-based models [18]. These models improve word-based models since they are able to translate sequences of words (phrases) and constitute the state-of-the-art in MT.

3.4.1. Word-based translation models

Brown et al. [21] approached the problem of MT in Eq. (1) from a statistical point of view as a search problem of a translation \( t \). In this approach a hidden variable \( a \) is introduced that represents the alignment between the words in the source and target sentence. In this way, an alignments \( a \) is defined as a vector of length \( |t| \), in which the \( i \)-th element corresponds to the source position \( j \), i.e., the word \( s_j \), to whom \( t_i \) is aligned. Formally, we can model the posterior probability of the target sentence \( t \) being a translation of the source sentence \( s \) by marginalizing over the set of all possible alignments between the words in \( t \) and the words in \( s \),

\[
Pr(t \mid s) = \sum_a Pr(t, a \mid s)
\]

Then, \( Pr(t, a \mid s) \) can be decomposed using the chain rule. After taking some strong assumptions, two distributions are obtained. First, the alignment model, \( Pr(j \mid i, |s|) \), represents the probability of the target word \( t_i \) to be aligned with the source word \( s_j \) for a source sentence of length \(|s|\). Second, the word translation model, \( Pr(t_i \mid s_j) \), models the probability of the target word \( t_i \) to be a translation of the source word \( s_j \). The above assumptions are necessary to make model estimation tractable and result in the so-called model 2 (M2) [21].

In M2, the alignment probability, \( Pr(j \mid i, |s|) \), can be approximated by the relative frequency of position \( j \) in the source sentence to be aligned with position \( i \) in the target sentence for a source sentence of length \(|s|\). On the other hand, the translation probability, \( Pr(t_i \mid s_j) \), can be approximated by a word-to-word statistical dictionary which essentially is the relative frequency of \( t_i \) being aligned with \( s_j \). Nonetheless, these frequencies cannot be estimated directly since the real alignments are unknown. Thus, the EM algorithm is needed to reliably estimate these probabilities [21]. Model 1 (M1) is a particular case of word-based models where the alignment probability is approximated by an uniform probability distribution, \( Pr(j \mid i, |s|) \approx (|s| + 1)^{-1} \).
Returning to our original problem, we can approach $Pr(d \mid p, s)$ in Eq. (7) with word-based translation models with some assumptions. First, from the prefix $p$ we can obtain the position of the erroneous word to be corrected, $i = |p| + 1$ ignoring the rest of the words in the prefix,

$$Pr(d \mid p, s) \approx Pr(d \mid i, s) \tag{9}$$

Then, we can introduce the alignment between $d$ and the words from the source sentence by summing for every possible position $j$ in $s$,

$$Pr(d \mid i, s) = \sum_{j=1}^{|s|} Pr(d \mid j, i, s) = \sum_{j=1}^{|s|} Pr(j \mid i, s)Pr(d \mid j, i, s) \tag{10}$$

Finally, if we assume, in a similar way to M2, that $Pr(j \mid i, s)$ does not depend on $s$ but on $|s|$, and that $Pr(d \mid j, i, s)$ does not depend on the whole $s$ but just on the source word $s_j$ aligned to $d$, then we can approximate Eq. (10) as

$$Pr(d \mid i, s) \approx \sum_{j=1}^{|s|} Pr(j \mid i, |s|)Pr(d \mid s_j) \tag{11}$$

where $Pr(j \mid i, |s|)$ is an M1 or M2 alignment model and $Pr(d \mid s_j)$ is a statistical dictionary.

To clarify the role of the alignments and the dictionary, observe Fig. 4. The source sentence is shown in the middle. Each word has its corresponding position, $j$, as a subscript. Above each word, there is a list of its most probable translations using the dictionary. Grey levels are proportional to the probability of the dictionary. On the other hand, in the bottom, there is a possible translation, which has an error in position $i = 4$. Below that, the user is trying to correct that mistake by introducing the word $\cup$. Each link between a source word and the target word in position 4 represents the alignment probability. The stroke boldness is proportional to the M2 alignment probability.

Note that for an M1 model, all alignments would have had the same thickness.

If we focus on the possible candidate transcriptions of $\cup$, we realize that there are two possibilities that could create confusion to the decoder: ‘if’ as translation of ‘si1’ and ‘in’ as translation of ‘ens’ due to the fact that the
strokes for ‘is’, ‘if’ and for ‘in’ can be very similar. Both can compete with the correct transcription ‘is’. The first, has a high probability in the dictionary, \( Pr(\text{if} | \text{si}_1) = 0.88 \), whereas \( Pr(\text{is} | \text{se}_5) = 0.46 \), \( Pr(\text{is} | \text{en}_{\text{a}_8}) = 0.34 \) and \( Pr(\text{in} | \text{en}_{\text{a}_8}) = 0.40 \). Then, since the M1 model has a uniform alignment probability, it would assign a higher probability to ‘if’ than to ‘is’. However, ‘si\(_1\)’ actually has a lower probability of being aligned with ‘not\(_4\)’. Therefore, the M2 model is able to solve this shortcoming thanks to the alignments with high probability to the correct words. In this case, \( Pr(5 | 4, 10) = 0.38 \) and \( Pr(6 | 4, 10) = 0.12 \), whereas \( Pr(1 | 4, 10) = 0.04 \).

### 3.4.2. Phrase-based translation models

Word-based translations provided a basis for MT. However, their performance regarding translation quality was not sufficient. Their limitation resides in that they cannot model properly context information [22]. Phrase-based models aim at reducing this problem by translating phrases (fragments of sentences) instead of single words. These models were popularized by Och and Ney [23], who established the state-of-the-art phrase-based log-linear models.

Phrase-based models offer a great opportunity to estimate \( Pr(d | p, s) \). However, we cannot use these models directly, as we did with word-based models. One limitation of phrase-based models is that their probabilities are ‘peaky’ and, usually, they cannot model all possible translations. As a result, it is possible that \( Pr(d | p, s) = 0 \) for a user established prefix like it would be the case in IMT. Then, it is necessary to smooth these probabilities. For instance, we can generate \( n \)-gram-like models from the hypotheses in a word graph (WG) of a MT system [24].

Word graphs contain a set of the most likely translations of the source sentence. They can encode a large number of translations in a more efficient way than \( n \)-best lists. Although one may think that the WG could be directly used, there are some details that must be taken into account. First, WGs do not contain all the possible translations since, in practice, many pruning techniques must be used to generate the translations efficiently. Second, phrase-based models are not good dealing with long distance alignments due to the introduction of heuristic length constrains, and thus, WGs do not present sentences with long distance reorderings. In those cases, a user validating a prefix \( p \) that is not contained in the WG would obtain a zero probability in \( Pr(d | p, s) \). Hence, it is interesting to smooth the probability distribution encoded in the WGs. To do so, WGs can be simplified in the way that language modeling is typically approached: we make each word to depend only on the preceding \( n - 1 \) words instead of depending on the whole prefix. As a result, Eq. (7) can be rewritten as

\[
\hat{d} \approx \arg \max_d Pr(d | p^{i-1}_{i-n+1}, s) Pr(x | d)
\]

where \( p^{i-1}_{i-n+1} \) are the words in the prefix from position \( i - n + 1 \) to position \( i - 1 \), i.e., \( Pr(d | p^{i-1}_{i-n+1}, s) \) only takes into account the latest \( n - 1 \) words from the prefix. Note that \( Pr(d | p^{i-1}_{i-n+1}, s) \) is very similar to a \( n \)-gram language model except for the dependency on \( s \). We used a similar approach for dictation of handwritten historical documents [25, 26] and speech interaction to IMT [13]. Khadivi and Ney [14] presented a closely related approach for generating \( n \)-gram-like models from \( n \)-bests lists instead of WGs. The advantage of the \( n \)-gram-like prefix modeling assumption is that the models only take into account a limited size of the history, and thus, can provide a smoother
probability distribution.

Formally speaking, a WG $L$ is a directed, acyclic, weighted graph with an initial node $q_i$ and a final node $q_f$. A link $l$ is defined as any edge between two nodes; each link has associated a begin node $b(l)$, an end node $e(l)$, a hypothesized word $w(l)$, and a score $f(l)$; each link can be considered as a hypothesis $w(l)$ between the nodes $b(l)$ and $e(l)$ with score $f(l)$. Any path from $q_i$ to $q_f$ forms a translation hypothesis $t$. In MT, $f(l)$ is the score of the log-linear phrase based model for that particular link. An example of WG is displayed in Fig. 5.

Let $Pr(U_q \mid s)$ be the posterior probability of all the paths that use the node $q$ and let $Pr(U_l \mid s)$ be the posterior probability of all the paths that use the link $l$. These probabilities can be efficiently computed with a forward-backward-based algorithm [27]. Then, the average counts of word sequences for a given source sentence can be estimated efficiently as in [28]. For a given $n$-gram length:

$$C^*(d_{i-n+1}^i \mid s) = \sum_{l_{i+n}^i \in \mathcal{N}(d_{i-n+1}^i)} \prod_{j=1}^n Pr(U_{l_{i+j}} \mid s) \prod_{j=2}^n Pr(U_{b(l_j)} \mid s)$$ (13)

where $\mathcal{N}(d_{i-n+1}^i)$ is the set of all the sequences of concatenated links in $L$ that produce the sequence of words $d_{i-n+1}^i$.

An example of such sets on a simplistic WG is shown in Fig. 5 for the 2-grams ‘feature cannot’ and ‘feature is’.

Then, $C^*(\text{feature cannot} \mid s)$ and $C^*(\text{feature is} \mid s)$ can be computed as

$$C^*(\text{feature cannot} \mid s) = \frac{0.4 \cdot 0.4}{0.4} = 0.4$$
$$C^*(\text{feature is} \mid s) = \frac{0.6 \cdot 0.24}{0.6} + \frac{0.6 \cdot 0.15}{0.6} + \frac{0.6 \cdot 0.21}{0.6} = 0.6$$

That is, ‘feature is’ appears 0.6 times in average in the possible set of translation, whereas ‘feature cannot’ only appears 0.4 times. Note that if a sequence of words appears more than once in a sentence, the average counts might exceed 1.

Now, $n$-gram-like probabilities from the WG with posterior probabilities can be calculated after a proper normal-
\[
Pr(d_i \mid d_{i-n+1}^{i-1}, s) = \frac{C^*(d_{i-n+1}^i \mid s)}{C^*(d_{i-n+1}^{i-1} \mid s)} \quad (14)
\]

Then, Eq. (14) can be used directly in Eq. (7) to approximate \(Pr(d \mid p, s)\). In other words, given a sequence of words \(d_{i-n+1}^{i-1}\), \(Pr(d_i \mid d_{i-n+1}^{i-1}, s)\) can be estimated by summing up the posterior probabilities of all sentences containing the sequence \(d_{i-n+1}^{i-1}\).

The estimation in Eq. (14) presents the problem that many \(n\)-grams are not seen in the WG. Then, they will have zero probability, and the HTR system will fail to recognize them. A common approach is to rely on simpler models to account for unseen events using back-off models [29]. As the estimated counts are not real counts (they vary from 0 to the number of times the \(n\)-gram occurs in a sentence), typical discount methods cannot be applied [30]. However, absolute discount can be used [31], which consists in subtracting a constant, \(\epsilon\), from \(C^*\).

Furthermore, only words present in the WG are included into the model (which implies a high number of out-of-vocabulary words (OOV), since WGs only contain the words of the most likely hypotheses). The OOV problem is solved by distributing the discounted mass from the unigram among the remaining words of the vocabulary.

Finally, to improve the estimation of unseen events, \(n\)-grams from the WG can be interpolated linearly with the standard \(n\)-gram model:

\[
Pr_\gamma(d \mid p, s) = \gamma Pr(d \mid p, s) + (1 - \gamma) Pr(d \mid p) \quad (15)
\]

This way, the words that were not used by the MT engine are assigned a meaningful probability.

3.5. Integrated HTR and IMT decoding

Previous models assume a two-step process, in which the strokes are first decoded into a word or phrase, and then, the decoded word is used to correct the output of the IMT system. However, this decoding can be performed in an integrated way by marginalizing over every possible decoding \(d\) in Eq. (2):

\[
\hat{t} = \arg\max_t \sum_d Pr(t, d \mid p, x, s) \quad (16)
\]

Note that Eq. (16) sums over all possible values of \(d\), but we also are interested in the result of the decoding. Then, we can decompose Eq. (16) using the chain rule. Approximating the sum by the maximum, and assuming that \(Pr(t \mid p, x, d, s)\) does not depend on \(x\) if \(d\) is known,

\[
\hat{d} \approx \arg\max_d Pr(d \mid p, x, s)Pr(t \mid p, d, s) \quad (17)
\]

where \(\hat{d}\) can be obtained as a byproduct of the decoding of \(\hat{t}\).

The first term in Eq. (17) can be approximated as in Eq. (3), Eq. (4), Eq. (5), Eq. (11) or Eq. (12). The second term is a prefix conditioned translation model as in Eq. (2). This probability forces \(d\) not just to be a good translation of \(s\) but to form part of a sentence that is good translation of it. Hence, the decoding of \(d\) is benefiting from a new source of information.
4. Experiments

In this section, we present a set of experiments to assess the performance of the MT specific HTR systems described in the above sections. Two kinds of experiments were conducted. First, the word-based experiments assume that the user only writes one word at a time. Second, in the phrase-based experiments the user writes a set of consecutive erroneous words. Additionally, two corpora were generated from the Xerox corpus, one with Spanish phrases from translations of English sentences and the other one with English phrases from translations of Spanish sentences. The details of how the two corpora were generated are given in Sec. 4.3.

4.1. IMT corpus: Xerox

The Xerox corpus, created in the TT2 project [32], was used for the experiments, since it has been extensively used in the literature to evaluate IMT systems. It consists of a collection of technical manuals in English, Spanish, French, and German. The English version is the original document, while the others are professional translations of the original. The English and Spanish versions were used in the experiments. The training data was used to generate the translation models. Examples of sentence pairs are shown in Fig. 6. The corpus consists of $56k$ sentences of training and a development and test sets of $1.1k$ sentences. The development set was used to find the tuning parameters that were used in test. Test perplexities for Spanish and English are 35 and 51, respectively. In addition, the Spanish test set has 0.7% out-of-vocabulary running words, whereas the English test set has 0.6% out-of-vocabulary running words.

4.2. HTR corpus: Unipen

For on-line HTR, the UNIPEN corpus [33] was used. The training data was composed of symbols, digits and the 1000 most frequent English and Spanish words. The words were generated by concatenating different instances of characters from the same writer, with a total of 17 different writers. Overall, 68 character classes and a total of $23.5k$ unique character instances were used to generate all the $43.8k$ training samples. The feature extraction and modeling we used was based on Pastor et al. [2]. Basically, the strokes were preprocessed by eliminating pen-up points and consecutively repeated points. Then, a low pass filter was applied to reduce noise by replacing each point with the mean of its neighbors [3]. From the resulting trajectory, 6 features were extracted:

- the vertical position is normalized by scaling and translating it to $[0, 100]$ keeping aspect ratio.
- the first and second derivatives for the vertical and horizontal position.
- the curvature, which is the inverse of the radius of the curve in each point.

Next, these feature vectors were used to train the morphological models, which were represented by left-to-right continuous density Hidden Markov Models (HMM) [34] with Gaussian mixtures and variable number of states per character. Three users were separated from the training process to produce the words from concatenated characters.
Figure 6: Examples of paired sentences in Spanish and English extracted from the Xerox corpus.

<table>
<thead>
<tr>
<th>Spanish</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>use este botón para ampliar la búsqueda de dispositivos xerox.</td>
<td>use this button to expand the search for xerox devices.</td>
</tr>
<tr>
<td>la búsqueda puede ampliarse para incluir otros nombres de comunidades de snmp que se han agregado a la red.</td>
<td>the search may be expanded to include additional snmp community names that have been added to your network.</td>
</tr>
</tbody>
</table>

Figure 7: Examples of pen strokes from the UNIPEN database used for the simulation of HTR. The words were obtained by concatenating random character instances from the corresponding user.

<table>
<thead>
<tr>
<th>user</th>
<th>another</th>
<th>recursos</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>another</td>
<td>recursos</td>
</tr>
<tr>
<td>User 2</td>
<td>another</td>
<td>recursos</td>
</tr>
<tr>
<td>User 3</td>
<td>another</td>
<td>recursos</td>
</tr>
</tbody>
</table>

for the development sets, which were used to find the optimal tuning parameters, and test sets. Examples of generated word in Fig. 7.

4.3. Procedure

For the word-based experiments, the simulation of the user interaction was performed in the following way. First, the publicly available IMT decoder Thot [35] was used to run an off-line simulation for keyboard-based IMT. To do this, we translated each test source sentence. Then, we obtained the longest correct prefix comparing to the reference. Next, we took the word that followed that prefix as the word the user would introduce as a correction. Finally, we used the prefix, and the correct word to obtain a new translation. This was repeated until the reference was obtained. As a result, a list of words that the system failed to predict was obtained. Supposedly, this would be the list of words that the user would correct with handwriting.

Then, from UNIPEN corpus, three writers were selected to simulate the user interaction. For each writer and for each of the words in the list of corrections, 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 were decoded using the proposed systems with iAtros decoder [36]. The 3-gram perplexities for the generated words are 205 and 226.
for development and test, respectively, in Spanish, and 242 and 336 for English. It is worthy of note these high perplexities, when for the whole dev and test sets the perplexities are 35 and 51. The word lists were extracted from the erroneous translations that were generated with a decoder using the very same n-grams models used to compute the perplexity. Hence, it is reasonable to assume that if the decoder failed to translate these words it was in part because the language probabilities were low enough, i.e., these probabilities were not well estimated, resulting in a high perplexity. Finally, the number of words in the development sets are 2767 for Spanish and 2398 for English, and in the test sets 2248 and 2102, respectively.

For the phrase-based experiments, the development and test sets were constructed in a similar way. In this case, from the word lists aforementioned, we concatenated the strokes of the words that were consecutive in the original text to form strokes of phrases. For instance, if the MT system had translated ‘lista de impresoras’ to ‘list of printers’ when the user preferred ‘printer list’, in the word-based scenario we would have generated the word ‘printer’ and the word ‘list’. In the phrase-based scenario, as both errors are consecutive, we would have concatenated them in a single phrase as ‘printer list’. Figure 8 illustrates a box-and-whisker diagram of the phrase lengths in the different sets. We can observe from the whiskers that the majority of the phrases are less than 3 words for Spanish and 6 for English, whereas for the outliers the lengths reach a maximum at 12 and 18, respectively. Note, however, that the interquartile range is between 2 and 3, meaning that half of the phrases are reasonably short. Finally, the number of phrases in the development sets are 941 for Spanish and 896 for English, and in the test sets 1268 and 1130, respectively.

4.4. Evaluation measures

The performance of the word-based HTR system has been assessed with the classification error rate (CER). CER is the ratio between the number of misrecognized words and the total number of words. On the other hand, the phrase-based HTR system has been assessed with the word error rate (WER), which can be computed as the number of substitutions, deletions and insertions needed to transform the hypothesis into the reference, normalized by the
number of words in the reference. The results present the average error of the three users.

4.5. Results

In this section, we will compare the performance of the proposed systems. In order to make the references easier, we will name the different systems as follows:

**HTR.** The baseline HTR system as defined in Eq. (3).

**ERR.** The baseline HTR system after removing the erroneous word, Eq. (4).

**nPREF.** In Eq. (5), the latest \( n \) words of validated prefix in the target sentence are taken into account.

**M1.** In Eq. (11), information regarding the dictionary is used, but the alignment probabilities are uniform.

**M2.** In Eq. (11), the dictionary and the alignment probabilities are used.

**nWG.** In Eq. (12), the system uses an \( n \)-gram that has been extracted from the translation WG.

Furthermore, if the decoding is performed in an integrated way, the system will be marked with \( +\text{IMT} \). Besides, several of the proposed systems can be combined by linear interpolation as in Eq. (15). In this case, we will use \(+\) symbol to mark which models were interpolated. The interpolation parameters were obtained in the development set to optimize the accuracy.

In addition, the proposed language models were encoded as \( n \)-grams. The aim of this is two-folded. First, we would like to leverage current HTR systems without custom software modifications. Second, since the new sources of information are added early in the HTR system, we expect to reduce the error cascade produced in post-processing error correcting systems. However, although all the proposed models can be trivially encoded as 1-grams for the case of word-based recognition, some of them cannot be encoded efficiently for \( n \)-grams as such and require special search algorithms. As these cases are out of the scope of the current paper, such models will not be evaluated for phrase recognition. Nevertheless, these models could also be applied in a post-processing rescoring stage. For instance, both \( M1 \) and \( M2 \) models can be easily encoded as a 1-gram for word-based recognition. As there is just one possible value for \( i \) and \( s \), the 1-gram can be built by computing Eq. (11) for each word of the vocabulary. In contrast, \( M2 \) models cannot be encoded as \( n \)-grams for phrase recognition since the probability depends on the position \( i \) of the hypothesized word, and then, \( i \) should be stored in the search algorithm for every word hypothesis. Luckily, \( M1 \) models assume independence of the position \( i \) so they can be encoded as a 1-gram even for the case of phrase recognition.

Finally, as it is typical in modern HTR and IMT models, the different probability distributions must be scaled, particularly the language model. Here, the optimum language model scaling factor, \( \lambda \), was chosen to optimize the average CER or WER in the development set of the three writers with the downhill simplex method [37]. There were
not significant differences in the optimum parameters obtained separately for each writer. Therefore, the estimation of these parameters seems rather robust to the variability of writers.

Regarding the results for the word-based experiments, Fig. 9 shows the test CER for different values of $\lambda$ for the most relevant systems. First, it must be pointed out that the optimum $\lambda$ from the development set approximated quite well the test optimum, i.e., the estimation of $\lambda$ does not present much overfitting. The only exception was the 2WG system for which an extra error reduction of 0.5% absolute points could have been achieved.
Second, we should note the effect of adding ERR to the system on the error rate. A small improvement can be noticed in Spanish. However, the curves in English overlap. The explanation for this is a bit involving. Note that Spanish is a more inflected language than English. For example, ‘both’ (in English) can be translated by ‘ambos’ or ‘ambas’ (in Spanish), depending on the gender, and having very similar writings. In contrast, ‘añade’ (in Spanish) can be translated by ‘adds’ (in English). Thus, we can see how translating from a less inflected language to a more inflected language introduces extra ambiguity. Furthermore, the possible translations of ‘both’ present also a similar spelling. Conversely, the ambiguity is reduced in the opposite direction. Table 1 shows the 5-best list of the HTR scores for the words ‘ambos’ and ‘adds’. In the first case, ‘ambas’ and ‘ambos’ are the two most likely words in the HTR system, which differ in just one character and have similar HTR scores. Now, imagine that the MT engine mistranslates ‘both’ to ‘ambas’, by changing the gender of the word. Then, by saying that ambas is not correct with the ERR model, we give the system the opportunity to amend the error himself. However, in the English case, none of the words are synonyms of the word to recognize, and thus is more difficult to find the mistranslated word at the top of the n-best list. As a consequence, it is very unlikely that ERR achieves much improvement when translating from Spanish to English.

With respect to the nPREF models, only 4PREF has been displayed in the plots. The improvement over the baseline is consistent and significant. The experiments were run on 2PREF, 3PREF and 5PREF as well. However, only 2PREF for English performed slightly worse than 4PREF. Longer prefixes achieved almost the same performance.

With respect to the systems using the translation models in Fig. 9c and Fig. 9d, we can see that these systems usually outperform the best basic system, 4PREF+ERR. The exception for this is 2WG for English, which shows a small performance degradation with respect to 4PREF+ERR. Still, 2WG systems do not seem to improve the basic systems significantly. Although several nWG systems were tested, any of them showed improvements over 2WG. On the other hand, M2 systems achieve good improvements, although they are simpler than 2WG. A reason for that is that M2 models have a smoother distribution probability and nWG systems need some sort of hypothesis pruning.
Table 2: Summary of the CER results for word-based recognition. The results show various language modeling approaches for the test sets. In **boldface** the best systems.

<table>
<thead>
<tr>
<th>System</th>
<th>Spanish</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTR</td>
<td>11.1</td>
<td>9.9</td>
</tr>
<tr>
<td>4PREF+ERR</td>
<td>9.9</td>
<td>9.5</td>
</tr>
<tr>
<td>2WG+ERR</td>
<td>9.8</td>
<td>9.4</td>
</tr>
<tr>
<td>M1+ERR</td>
<td>9.4</td>
<td>9.0</td>
</tr>
<tr>
<td>M2+ERR</td>
<td>8.6</td>
<td>7.7</td>
</tr>
<tr>
<td>2WG+4PREF+ERR</td>
<td>9.2</td>
<td>7.9</td>
</tr>
<tr>
<td>M2+4PREF+ERR</td>
<td>9.0</td>
<td>7.5</td>
</tr>
<tr>
<td>2WG+4PREF+ERR+IMT</td>
<td>9.2</td>
<td>7.9</td>
</tr>
<tr>
<td>M2+4PREF+ERR+IMT</td>
<td>8.9</td>
<td>7.5</td>
</tr>
<tr>
<td>ALL</td>
<td>8.9</td>
<td>7.4</td>
</tr>
</tbody>
</table>

In fact, the average number of candidates with probability greater than zero is 292 for M2 while it is 38 for 4WG. IMT suffer even more from this problem with 2 candidates average.

A summary of the different alternatives studied for the word-based experiments is shown in Table 2. First, with only the basic information, 4PREF+ERR clearly outperforms HTR. Second, using translation models we can achieve further improvements. Since M2 performs much better than M1 we can deduce that alignment information is crucial for the translation models. On the other hand, 4WG performance is worse than word-based translation models. As it has been explained before, that might be due to the poorly smoothed probability distribution. Another reason might be that, in the process of obtaining n-gram models, information regarding alignments is lost as a result of the n-gram assumptions. When interpolating with 4PREF, M2 models do not show significant improvements. In fact, for Spanish, the system presents over-fitting, since performance in development improves but in test decreases. However, 4PREF smoothes 2WG distribution achieving close results to word-based models. Next, by introducing IMT, small improvements can be obtained. Not surprisingly, IMT suffers from the same problems than 4WG, but even more prominent. Finally, including all systems we can observe the best results overall, except for the over-fitting in the Spanish test set. Thus, 2WG seems to contribute slightly to improve the final model accuracy.

Table 3 shows the WER for phrase-based recognition. First, it must be noted that the results for ERR, M2, and IMT are not shown, since they would require a different search engine. In addition, it is worth of mention that the baselines for phrase-based HTR have almost the double error rate than the word-based baselines. This is caused primarily because the segmentation for the words in the phrases are unknown. Then, it is the search algorithm that
must find the most likely segmentation. As a result, segmentation errors are propagated to word errors. If we look at
the results regarding the \( n \text{WG} \) models, they perform unexpectedly bad when used alone. However, when interpolated
with 3PREF they show a good improvement. As in word-based recognition, word-based translation models show the
best results, especially when interpolated with other models.

To sum up, all the proposed systems significantly outperform the baseline recognizer. Basic models obtain a
good improvement over the baseline. However, adding information from the translation may achieve remarkable
results. Although more complex translation models suffer from smoothing problems, they can also contribute when
interpolated with the rest of the models.

### 4.6. Error Analysis

An analysis (Table 4) of the results for the best word-based model shows that 49.2% to 54.4% of the recognition
events were produced by punctuation and other symbols. To circumvent this problem, we proposed a contextual
menu in [16]. With such menu, errors would have been reduced (best test result) to 4.4% in Spanish and 3.5% in
English. Out-of-vocabulary (OOV) words plus zero probability (P0) words (the words for which the decoder assigned
zero probability or were pruned out) also summed up a big percentage of the error (40.3% and 28.9%, respectively).
Finally, the rest of the errors were mostly due to one-to-three letter words, which can be basically a problem of
handwriting morphological modeling.

On the other hand, phrase recognition presents a different error distribution. First, note that two new classes of
events have been introduced: deletions and insertions. The former account for the words in the reference that have
been omitted, whereas the latter account for words inserted in the output hypothesis but do not correspond to any
word in the reference. Both contribute to generate hypotheses with lengths different to their respective references,
since HMM models are not able to perform an accurate segmentation. Then, as a result, the proportion of recognition

---

### Table 3: Summary of the WER results for phrase-based recognition. The results show various language modeling approaches for the test sets. In **boldface** the best systems.

<table>
<thead>
<tr>
<th>System</th>
<th>Spanish</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTR</td>
<td>16.8</td>
<td>18.6</td>
</tr>
<tr>
<td>3PREF</td>
<td>16.3</td>
<td>18.0</td>
</tr>
<tr>
<td>2WG</td>
<td>18.9</td>
<td>19.7</td>
</tr>
<tr>
<td>M1</td>
<td>17.0</td>
<td>17.4</td>
</tr>
<tr>
<td>2WG+3PREF</td>
<td>16.2</td>
<td>16.6</td>
</tr>
<tr>
<td>M1+3PREF</td>
<td><strong>15.2</strong></td>
<td><strong>15.5</strong></td>
</tr>
<tr>
<td>M1+2WG+3PREF</td>
<td>15.2</td>
<td>15.5</td>
</tr>
</tbody>
</table>

---
Table 4: Detailed analysis of the word-based and phrase-based recognition errors. Five classes have been identified to produce the most amount of recognition errors. The second column shows samples of misrecognized words for these classes. Columns three and four are the percentage of these classes among the total number of misrecognized words for Spanish (es) and English (en), respectively. Columns five and six are the percentages for the phrase-based experiments. In this case, the percentage of substitutions, insertions and deletions is also shown.

errors from the ‘others’ category increases from 3 to 20. In contrast, the proportion of errors regarding punctuation symbols decreases. Finally, it is to be remarked how the errors for short words have increased, probably because of small insertions or deletions.

4.7. Reducing Effort Correcting HTR errors

In case an HTR error is committed, the user may fall back to the virtual keyboard and type the correct word. The problem with this kind of keyboards is that typing is slow. To minimize this problem, we propose a contextual menu with a list of the $n$-best candidates (excluding the erroneous word). The aim is to reduce the number of clicks needed to obtain the correct word with respect to a conventional virtual keyboard. As a baseline, for each HTR mistake, we count the number of clicks needed to input the correct word as: one click to pop up the keyboard, plus the number of characters in the word, plus on click to close the keyboard. For the Spanish test set, the average number of clicks per word amounts to 9.3, while for English it is 9.1 for the best word-based models in Table 2. This values can be surprisingly high, since it is known that the average word length is 4.5, i.e. the average number of clicks per word 6.5. However, it must be noticed that longer words are also more difficult to recognize. Thus, the average word length in the erroneous words is higher.

If the contextual menu is used, we count: one click for opening the menu plus one for choosing a word. If the correct word cannot be found in the $n$-best list, then we add: one count for the keyboard, plus the number of characters, plus...
plus a closing click. In Fig. 10, we can see, on the left axis, the CER for a given size of the $n$-best list. Clearly, the error almost reduces to a quarter, around $n = 5$, with respect to the baseline. Between 10 and 15, the error stabilizes. Note that from 5 to 10 is still a reasonable amount of candidates to be shown in a circular menu. For more than 15, the CER almost equals the error for OOV+P0, since they cannot be found in $n$-best lists. On the right axis, we can observe the average number of clicks per word necessary to correct the mistakes. For $n = 1$ the number of clicks is reduced to 2.0. A trade-off can be found at $n = 7$ with 1.83 (80% relative improvement w.r.t. the baseline) and 1.82 (78% relative improvement), for Spanish and English, whereas the lower bounds are 1.75 and 1.73, respectively.

![Figure 10: Reduction of CER and number of clicks as a function of the $n$-best list size.](image)

5. Final Remarks

While the techniques addressed in this paper have been focused on correcting machine translation output, in reality some of them can be generalized to the correction of other automatically generated outputs. In particular, ERR and $n$PREF can be used to improve HTR accuracy for any tasks in which $n$-grams can be used for language modeling, e.g., [7]. Obviously, M1 and M2 are MT specific, but $n$WG can be used for many other structured prediction problems where a word graph can be generated as an output. In fact, in a similar way to this work, $n$WG has been successfully used for speech-enabled user interfaces for IMT [13] and for dictation of historical documents [25, 26]. In the same way, integrating HTR with interactive systems is possible for other applications as far as $n$WG is available. Nonetheless, using more specific techniques, such as M2, although less general, have proven to be more effective. Finally, we recommend that the integration of contextual information in the decoding is performed in an early stage of the decoding process, avoiding cascade errors. More importantly, if the techniques can be encoded as $n$-grams, as the techniques presented here, it will allow practitioners to improve their HTR systems without modifying their preferred HTR engines.
6. Conclusions and Future Work

In this paper we have described a task specific on-line HTR system to operate with an MT application. It is worth of note that all the proposed systems significantly outperform the baseline recognizer. Basic models obtain a good improvement over the baseline. However, translation models achieve remarkable results. Although more complex translation models suffer from smoothing problems, they also contribute when interpolated with the rest of the models. We also have introduced a new method for correcting HTR mistakes that consists on a contextual menu with the $n$-best candidates. The results show that a list with as few as 7 candidates allows to correct the HTR mistakes with just 1.83 clicks per word.

On the other hand, the analysis of the results has shown two important issues to be tackled. First, the system should be able to decode unknown words since they are a clear limitation to system performance. A solution for this might be to use character language models instead of word language models, a technique that has achieved promising results in other areas. Second, phrase-based models could benefit from better smoothing methods. Alignment information should be also taken into account more explicitly in these models. Furthermore, other alternatives could also be explored, as more advanced word-based translation models (such as HMM, M3, M4 or M5) that cannot be used as $n$-grams in phrase-based decoding. These models could be used instead in the rescoring of the HTR WGs. Finally, if the rescoring of WGs shows promising results, it would be interesting to directly implement the more advanced MT models into the HTR search algorithm.

Acknowledgements

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), from the EC (FEDER/FSE), and from the Spanish MEC/MICINN under the Active2Trans (TIN2012-31723) project. It is also supported by the Generalitat Valenciana under grant ALMPR (Prometeo/2009/01) and GV/2010/067.


560 Vicent Alabau received the Master degree in computer science from the Universidad Politécnica de Valencia, Valencia, Spain, in 2003. Since that moment he has been working in several projects for the “Pattern Recognition and Human Language Technology Group” on speech recognition and translation, machine translation, and multimodal interactive tools for pattern recognition.

561 Alberto Sanchis received the Diploma and the Ph.D. degrees in computer science from the Universitat Politècnica de València, Valencia, Spain, in 1997 and 2004, respectively. In 2000, he joined the Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, where he is currently serving as an Associate Professor. His main research interests include pattern recognition and its application to speech and handwriting recognition and machine translation. He is now leading a project on handwriting and speech transcription.

562 Francisco Casacuberta is a full professor in the Universitat Politècnica de València (UPV), Spain. He is also one of the leaders of the Pattern Recognition and Human Language Technology research group in the Instituto Tecnológico de Informática. He received his Ph.D. degree from the Universidad de Valencia in 1981. His research interests include the areas of syntactic pattern recognition, statistical pattern recognition, machine translation, speech recognition and machine learning. He is a member of the Spanish AERFAI Society and the IEEE Computer Society.
Attachment C

Task 2.3

Luis A. Leiva, Vicent Alabau, and Enrique Vidal.


Luis A. Leiva, Vicent Alabau, Enrique Vidal
ITI/DSIC, Universitat Politècnica de València
Camí de Vera, s/n – 46022 Valencia (Spain)
{luileito,valabau,evidal}@{iti,dsic}.upv.es

Abstract
We present a straightforward solution to incorporate text-editing gestures to mixed-initiative user interfaces (MIUIs). Our approach provides (1) disambiguation from handwritten text, (2) edition context, (3) virtually perfect accuracy, and (4) a trivial implementation. An evaluation study with 32 e-pen users showed that our approach is suitable to production-ready environments. In addition, performance tests on a desktop PC and on a mobile device revealed that gestures are really fast to recognize (0.1 ms on average). Taken together, these results suggest that our approach can help developers to deploy simple but effective, high-performance text-editing gestures.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User interfaces—Input devices and strategies, interaction styles

Author Keywords
Text Editing; Gestures; UI Prototyping; Pointing Devices

Introduction
Gesture-based interfaces provide the user with a direct, natural form of interaction. Together with the popularity of stroke-based devices (e.g., touchscreens, e-pens, styli, tabletops, or surfaces), accurate gesture recognition and suitable prototyping tools are becoming essential. Within
In this context, text-editing applications are increasingly being enhanced with gestures, specially those applications that follow a mixed-initiative user interface (MIUI) principle [2], i.e., those in which the user and the system collaborate efficiently (Figure 1). For instance, CueTIP [8], CATTI [7], and IMT [6] are recent examples of text-editing applications with MIUIs partially commanded by gestures. In these systems, the user iteratively refines some automatic system output (or hypothesis), by providing corrective feedback that the system leverages to produce a better hypothesis.

**Current Challenges**

This work aims to solve three open problems when editing text on MIUIs. First, gestures and handwritten text must be unambiguously differentiated. Otherwise, if a gesture is misrecognized as text (or vice versa), cascading errors are likely to happen, so a) the actual user intention would be wrongly captured by the application; therefore b) it would not be possible for the system to derive a correct response; which c) would cause frustration, as d) the user would need to amend the erroneous response and resubmit the previously intended gesture (or text correction) again. Second, it is notably important to ensure both low recognition errors and low recognition times, since productivity is extremely mandatory when operating a text-editing MIUI. In this regard, users are typically willing to accept error rates up to about 3% or less, before deeming the technology as "too encumbering" [3]. Finally, it is mandatory for the system to know the context of an issued gesture, i.e., the application need to know information from the text itself which the user is interacting with, at the word (or even character) level, in order to provide the user with suitable corrections. Thus, on a text-editing MIUI, gestures must be performed over the text being edited.

The above-mentioned open problems constrain the design of the gesture vocabulary. Moreover, each type of application has unique operations and therefore requires specialized gestures. Even more, gestures are limited both by human memory and user performance, and hence they must remain simple. Figure 2 illustrates these ideas. Inspired in part by the Marking Menus techniques [1], our approach, named MinGestures, is based on the fact that drawing lines (1D gestures) with a pointer device is a very simple task and really easy for users to perform, but it also should be very efficient for computers to recognize, since the proposed gestures are linearly separable. This paper therefore provides a well-defined balance to deploy text-editing gestures on MIUIs.

**Implementation**

We first tried to implement our baseline set of eight 1D gestures (Figure 3) using state-of-the-art recognizers, among which we chose [5, 9, 10] for being easily customizable. Concretely, only $1 [10]$ would partially fit our needs. Protractor [5] is a faster version of $1$, as a result of rotating gesture samples to their optimal indicative angle prior and during recognition, and therefore it is not appropriate to deal with our full set of 1D gestures. $P [9]$ is another version of $1$ in which gestures are treated as clouds of points, so it cannot differentiate gestures on the basis of direction.

As reported in Pilot Study, after incorporating some tweaks to $1$, overall recognition error was around 2%, which was encouraging but perhaps still insufficient for editing text on MIUIs. Therefore, considering the simplicity of MinGestures, we opted for implementing a customized parametric recognizer, since target gestures must fit an assumed model (straight lines). Otherwise, the strokes should be identified as handwritten text.
Figure 4 provides a graphical overview of a proposed gesture set based on MinGestures, which, from our experience developing MIUIs (e.g., [6, 7]), incorporates essential operations to handwritten text and, at the same time, adequately solves the three open problems discussed in the previous section.

<table>
<thead>
<tr>
<th>Label</th>
<th>Action</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Substitute</td>
<td>Lorem Ipsum</td>
<td>Lorem Ipsum</td>
</tr>
<tr>
<td>Merge</td>
<td>Lorem Ipsum</td>
<td>Lorem Ipsum</td>
</tr>
<tr>
<td>Delete</td>
<td>Lorem Ipsum</td>
<td>Lorem Ipsum</td>
</tr>
<tr>
<td>Insert</td>
<td>Lorem Ipsum</td>
<td>Lorem Ipsum</td>
</tr>
</tbody>
</table>

Figure 4: Sample set of interactive text-editing operations that can be developed with MinGestures.

Lorem Ipsum dolor sit amet.
Lorem Ipsum dolor sit amet.
Lorem Ipsum dolor sit amet.

Figure 5: Words’ bounding boxes are normalized in height. This allows the user to easily select (or draw over) them, but also to find the gesture context.

Preliminaries: Contextualizing Gestures

For each text segment being edited, word bounding boxes are normalized in height (Figure 5). These “virtual” bounding boxes will be used to accurately detect the word(s) the user is interacting with.

Let $P = \{s_1 \ldots s_p \ldots s_{|P|}\}$ be a sequence of $|P|$ strokes, where $s_p = \{(x_1, y_1) \ldots (x_n, y_n) \ldots (x_{|s_p|}, y_{|s_p|})\}$ are sequences of $|s_p|$ 2D points.

On the one hand, the centroid $c_p = \frac{1}{|s_p|} \sum_{n=1}^{|s_p|} s_p$ informs about the word being edited (Figure 5), by searching

$$j^* = \arg \min_j d(c_p, c_j)$$

(1)

where $d(c_p, c_j)$ is the distance between the stroke centroid and the $j$-th bounding box centroid.

On the other hand, the angle $\theta_p = \tan^{-1} \frac{y_{|s_p|} - y_1}{x_{|s_p|} - x_1}$ measures the slope of the fitted line (if any). As shown in Figure 6, MinGestures uses a tolerance of $\epsilon_1 = 35^\circ$ for diagonal lines and $\epsilon_2 = 10^\circ$ for horizontal/vertical lines, although both parameters are customizable.

Disambiguating Gestures & Handwritten Text

Using our parametric approach, we tried Pearson’s $\rho$ as a discriminative feature, as suggested by Li and Hammond [4]. This feature, albeit being intuitive for this task, did not work for us, as indicated in the next section. In contrast, we can use a couple of stroke features that fit better this task. These features rely on the gestures lying on the $x$-axis. Thus, each feature must be rotated by its indicative angle, as in [5, 9, 10]. First, the aspect ratio

$$\varphi = \frac{w}{h}$$

where $w$ and $h$ are, respectively, the width and height of the stroke bounding-box, informs about the shape of a stroke, therefore “thin” strokes are likely to be near-straight lines. Second, the cummulative horizontal negative derivative

$$\Delta_\varphi = \sum_{n=2}^{|s_p|} \max(x_{n-1} - x_n, 0)$$

(3)

informs about points being drawn backwards; therefore if a submitted stroke gives $\Delta_\varphi \approx 0$, then it is monotonous in the (rotated) $x$-axis and therefore is likely to be a line.

Using these features, we found out that gestures can be easily disambiguated from handwritten text. Then, together with the taxonomy shown in Table 1, gestures are properly contextualized and potential collisions are solved. For instance, the character "l", a comma, or a dash,
could be misrecognized as lines, for which the context of bounding boxes adequately solves these ambiguities.

Recapitulation: Recognizer Workflow
Whenever the user stops writing on the UI (e.g., after some milliseconds of inactivity), the submitted pen strokes are inspected according to Equations (2) and (3), together with the following taxonomy.

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>Gesture labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>in</td>
<td>Substitute, Merge</td>
</tr>
<tr>
<td>in</td>
<td>out</td>
<td>Substitute</td>
</tr>
<tr>
<td>out</td>
<td>in</td>
<td>[unassigned]</td>
</tr>
<tr>
<td>out</td>
<td>out</td>
<td>Delete, Insert, Split, Validate, Undo, Redo</td>
</tr>
</tbody>
</table>

Table 1: Taxonomy of implemented gestures (Figure 4), based on the position of the first and last stroke points with respect to a word bounding box.

Figure 6: Accommodating gesture variability. [6a] The angular tolerances $\epsilon_1, \epsilon_2$ are user-customizable. [6b] Gestures are drawn on non-overlapping areas, so they can be robustly distinguished.

Then, if the first stroke is considered to be a line, the stroke angle is computed to classify the corresponding operation. Otherwise, the user would be substituting (handwriting) a word, in which case the strokes must be decoded by a handwriting recognition engine.

Pilot Study
We recruited 32 e-pen users (6 females) aged 26–36. All subjects were unpaid volunteers. A Wacom Bamboo ‘Pen & Touch’ digitizer tablet was used as input device in a regular PC (2 GHz CPU, 1 GB RAM) equipped with Ubuntu Linux. Participants were asked to perform each gesture up to 10 times, over the same mock-up sentence. Gestures were presented randomly, in order to avoid possible biases in learnability.

We also collected qualitative data about usage experience: Q1: Gestures are easy to perform; Q2: Gestures are easy to remember; Q3: Gestures do suffice for text-editing purposes; Q4: I am satisfied with this gesture recognizer. Questions were punctuated in a 1–5 Likert scale (1: completely disagree, 5: completely agree). Users were also encouraged to give free comments and ideas via an online survey at the end of the test.

Recognition Errors
We conveyed a series of experiments to assess how the recognizers performed in terms of accuracy and efficiency. The goal was to classify a given pen stroke into one of nine classes: each of the eight directions and handwriting text. As previously pointed out, first we should tell gestures and non-gestures apart.

Previously we have discussed some stroke features that can be used to detect lines. Nevertheless, we need to establish a threshold after which gestures and non-gestures can be discriminated. In order to identify the threshold without a bias, we decided to split the original corpus into two datasets. The training set consists of 5 samples per gesture plus text (9 classes) per user, with a total of 1440 samples. This set was used to obtain the threshold that minimizes the recognition error rate. The remaining samples (1351 in total) were used as an independent test set. The threshold used for these experiments was the one selected in the training phase. Finally, as the $1$ recognizer needs templates to operate, it was fed with a set of ‘perfect’ line samples in each of the eight directions.

Our initial approach was to use $1$ out-of-the-box. However, as this recognizer rotates all the gestures to its indicative angle, all lines were rotated to the vertical line. Moreover, $1$ scales gestures to a 1:1 aspect ratio, so lines become almost dots. Thus, results turned out to be random. Therefore, we decided to remove these limitations from $1$. We found that if a gesture was
recognized with less than 0.44 posterior probability, then an optimum was obtained with 2.22% error rate.

Next, we experimented with MinGestures (MG for short). Our first approach was to use Pearson’s $\rho$. Unexpectedly, $\rho$ proved to be not very robust for identifying lines, with an error of 6.94% with gestures being recognized as such if $\rho \geq 0.15$, which is very low (Figure 7a). Hence, we decided to rotate the gestures to lie down on the $x$-axis, and then compute the number of pixels drawn backwards (Equation 3). This resulted in a much better approach with an error of 0.35%, considering as non-gestures more than 1 pixels being drawn backwards (Figure 7b). Also, using the bounding box aspect ratio achieved very good results (0.14% with lines having an aspect ratio $\geq 3.4$, Figure 7c). Finally, aiming at an error-free recognizer, we combined the last two techniques. Indeed, we achieved a perfect recognizer (both in training and test) when a gesture is at least 3:1 with no more than 6 pixels drawn backwards. A summary of the results for both training and test is shown in Table 2. As it can be seen, eventually MinGestures behaves as a deterministic error-free interface.

Performance Evaluation
Firstly, we analyzed the time that users invested to draw the gestures. On average, they took 800 ms (SD=610). However, notice that the Substitute gesture penalized slightly these results, since users submitted unconstrained handwriting words during the test. Secondly, we computed the average time required to recognize each gesture with MinGestures (Table 3). We ran our recognizer 10 times over all samples in an traditional PC (an i686 with 2 GHz CPU and 2 GB of RAM). The PC needed 0.1 ms on average (SD=0.01) to recognize all gestures. To better understand the impact of the time performance, we repeated the same experiment on an HTC Nexus One running Android 2.2. The mobile device needed on average 0.11 ms (SD=0.43) to recognize all submitted gestures. Regarding the PC performance, this difference of 0.01 ms could be considered statistically significant $[\chi^2_{(1, N=2791)} = 4.66, p = .03]$. Nonetheless, in practice users would not complain between using MinGestures on a mobile device or on a traditional PC in terms of performance, given the narrow margin of difference. These results concluded that the proposed set of gestures are effortless to draw and really fast to recognize.

Qualitative Results
Regarding the four qualitative questions asked at the end of the acquisition tests, we observed that people liked MinGestures overall (see Table 4). We concluded that our recognizer is a convenient approach to deploy text-editing gestures on MIUs.

Limitations
The simplicity of our approach leads to a few inevitable drawbacks. First, MinGestures is suited to maximize accuracy and runtime efficiency. For that reason, this recognizer is domain-specific and could not fit a researcher’s needs in other applications. Thus, text processing applications, such as post-editing interfaces, or transcription and translation systems, are our main and only (although relatively wide) target.

Second, MinGestures provides at most $8 \times 2 \times 4 = 64$ gestures [directions, (un)touching a word, and inside/outside words’ bounding boxes], a set of actions that, however, should be enough for text-editing MIUs. Some guidance to implement more gestures could be differentiating them on the basis of time or speed. If needed, multistrokes gestures could be implemented by...
combining the core set of MinGestures with finite state automata. In any case, there is an inherent limitation of all user-independent systems: creating custom gestures is restricted to the set of primitives used in MinGestures.

All in all, although a concise recognizer like ours may not rival other systems in terms of power, flexibility, or complexity, it is our belief that it may be well suited for a wide range of devices such as tablets, surfaces, or handhelds computers.

Conclusion and Future Work

Stroke-based MIUIs devoted to create or modify text can be easily enhanced with simple gestures, without resorting to complex techniques or using recognizers that are too general. We stressed this fact and developed a deterministic approach to disambiguate among simple gestures and handwritten text, with runtime efficiency as primary focus. This paper may thus serve as a reference guide prior to designing and evaluating text-editing MIUIs.

For future work, we have devised two research avenues. On the one hand, we plan to incorporate more expressivity to MIUIs that are driven by MinGestures. For instance, multiple gesture strokes could be submitted together with handwritten text at a time, speeding thus the cooperative (and interactive and predictive) workflow carried out by the user and the system on a text-editing MIUI.

We already have started working on the integration of MinGestures in a production-ready machine translation system. In the context of the CasMaCat project\(^1\), our gesture set will assist professional translators to post-edit text interactively. The whole machine translation system is expected to be formally evaluated in a few months. It is our belief that MinGestures may enable a natural and accurate (interactive) text edition well beyond e-pen or touch devices.

Acknowledgements

This work is supported by CasMaCat Project 287576 (FP7 ICT-2011.4.2).

References


---

\(^1\)http://www.casmacat.eu
Attachment D

Task 2.4

Jesús González-Rubio, Daniel Ortiz-Martínez, José-Miguel Benedí, and Francisco Casacuberta.

Interactive machine translation using hierarchical translation models.

Interactive Machine Translation using Hierarchical Translation Models

Jesús González-Rubio, Daniel Ortiz-Martínez, José-Miguel Benedí, Francisco Casacuberta
D. de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de Vera s/n, 46021 Valencia (Spain)
{jegonzalez,dortiz,jbenedi,fcn}@dsic.upv.es

Abstract

Current automatic machine translation systems are not able to generate error-free translations and human intervention is often required to correct their output. Alternatively, an interactive framework that integrates the human knowledge into the translation process has been presented in previous works. Here, we describe a new interactive machine translation approach that is able to work with phrase-based and hierarchical translation models, and integrates error-correction all in a unified statistical framework. In our experiments, our approach outperforms previous interactive translation systems, and achieves estimated effort reductions of as much as 48% relative over a traditional post-editing system.

1 Introduction

Research in the field of machine translation (MT) aims to develop computer systems which are able to translate between languages automatically, without human intervention. However, the quality of the translations produced by any automatic MT system still remain below than that of human translation. Typical solutions to reach human-level quality require a subsequent manual post-editing process. Such decoupled post-translation solution is rather inefficient and tedious for the human translator. Moreover, it prevents the MT system from taking advantage of the knowledge of the human translator and, reciprocal, the human translator cannot take advantage of the adapting ability of MT technology.

An alternative way to take advantage of the existing MT technology is to use them in collaboration with human translators within a computer-assisted translation (CAT) or interactive framework (Isabelle and Church, 1998). The TransType and TransType2 projects (Foster et al., 1998; Langlais and Lapalme, 2002; Barrachina et al., 2009) entailed an interesting focus shift in CAT technology by aiming interaction directly at the production of the target text. These research projects proposed to embed an MT system within an interactive translation environment. This way, the human translator can ensure a high-quality output while the MT system ensures a significant gain of productivity. Particularly interesting is the interactive machine translation (IMT) approach proposed in (Barrachina et al., 2009). In this scenario, a statistical MT (SMT) system uses the source sentence and a previously validated part (prefix\(^1\)) of its translation to propose a suitable continuation. Then the user finds and corrects the next system error, thereby providing a longer prefix which the system uses to suggest a new, hopefully better continuation. The reported results showed that IMT can save a significant amount of human effort.

Barrachina et al., (2009) provide a thorough description of the IMT approach and describe algorithms for its practical implementation. Nevertheless, we identify two basic problems for which we think there is room for improvement. The first problem arises when the system cannot generate the prefix validated by the user. To solve this problem, the authors simply provide an ad-hoc heuristic error-correction technique. The second problem is how the system deals with word reordering. Particularly, the models used by the system were either mono- ordering. Particularly, the models used by the system were either mono-
tonic by nature or non-monotonic but heuristically
defined (not estimated from training data).

We work on the foundations of Barrachina et al., (2009) and provide formal solutions to these two challenges. On the one hand, we adopt the statistical formalization of the IMT framework described in (Ortiz-Martínez, 2011), which includes a stochastic error-correction model in its formalization to address prefix coverage problems. Moreover, we refine this formalization proposing an alternative error-correction formalization for the IMT framework (Section 2). Additionally, we also propose a specific error-correction model based on a statistical interpretation of the Levenshtein distance (Levenshtein, 1966). These formalizations provide a unified statistical framework for the IMT model in comparison to the ad-hoc heuristic error-correction methods previously used.

In order to address the problem of properly deal with reordering in IMT, we introduce the use of hierarchical MT models (Chiang, 2005; Zollmann and Venugopal, 2006). These methods provide a natural approach to handle long range dependencies and allow the incorporation of reordering information into a consistent statistical framework. Here, we also describe how state-of-the-art hierarchical MT models can be extended to handle IMT (Sections 3 and 4).

We evaluate the proposed IMT approach on two different translation task. The comparative results against the IMT approach described by Barrachina et al., (2009) and a conventional post-edition approach show that our IMT formalization for hierarchical SMT models indeed outperform other approaches (Sections 5 and 6). Moreover, it leads to large reductions in the human effort required to generate error-free translations.

2 Statistical Framework

2.1 Statistical Machine Translation

Assuming that we are given a sentence \( s \) in a source language, the translation problem can be stated as finding its translation \( t \) in a target language of maximum probability (Brown et al., 1993):

\[
\hat{t} = \arg \max_t \Pr(t \mid s)
\]

\[= \arg \max_t \Pr(t) \cdot \Pr(s \mid t)
\]

where \( \Pr(t \mid s) \) is the language model probability \( \Pr(t) \) that represents the well-formedness of \( t \) (\( n \)-gram models are usually adopted), and the (inverted) translation model \( \Pr(s \mid t) \) that represents the relationship between the source sentence and its translation.

In practice all of these models (and possibly others) are often combined into a log-linear model for \( \Pr(t \mid s) \) (Och and Ney, 2002):

\[
\hat{t} = \arg \max_{\mathbf{t}} \left\{ \sum_{n=1}^{N} \lambda_n \cdot \log(f_n(t, s)) \right\}
\]

where \( f_n(t, s) \) can be any model that represents an important feature for the translation, \( N \) is the number of models (or features), and \( \lambda_n \) are the weights of the log-linear combination.

2.2 Statistical Interactive Machine Translation

Unfortunately, current MT technology is still far from perfect. This implies that, in order to achieve good translations, manual post-editing is needed. An alternative to this decoupled approach (first MT, then manual correction) is given by the IMT
paradigm (Barrachina et al., 2009). Under this paradigm, translation is considered as an iterative left-to-right process where the human and the computer collaborate to generate the final translation.

Figure 1 shows an example of the IMT approach. There, a source Spanish sentence \( s = "\)Para ver la lista de recursos\) is to be translated into a target English sentence \( t \). Initially, with no user feedback, the system suggests a complete translation \( t_s = "To view the resources list\)”. From this translation, the user marks a prefix \( p = "To view\) as correct and begins to type the rest of the target sentence. Depending on the system or the user’s preferences, the user might type the full next word, or only some letters of it (in our example, the user types the single next character “a”). Then, the MT system suggests a new suffix \( t_s = "\)list of resources\) that completes the validated prefix and the input the user has just typed \( (p = "To view a\)”). The interaction continues with a new prefix validation followed, if necessary, by new input from the user, and so on, until the user considers the translation to be complete and satisfactory.

The crucial step of the process is the production of the suffix. Again decision theory tells us to maximize the probability of the suffix given the available information. Formally, the best suffix of a given length will be:

\[
\hat{t}_s = \arg \max_{t_s} \Pr(t_s | s, p)
\]

which can be straightforwardly rewritten as:

\[
\hat{t}_s = \arg \max_{t_s} \Pr(p, t_s | s)
\]

\[
= \arg \max_{t_s} \Pr(p, t_s) \cdot \Pr(s | p, t_s)
\]

Note that, since \( p t_s = t \), this equation is very similar to Equation (2). The main difference is that now the search process is restricted to those target sentences \( t \) that contains \( p \) as prefix. This implies that we can use the same MT models (including the log-linear approach) if the search procedures are adequately modified (Och et al., 2003). Finally, it should be noted that the statistical models are usually defined at word level, while the IMT process described in this section works at character level. To deal with this problem, during the search process it is necessary to verify the compatibility between \( t \) and \( p \) at character level.

### 2.3 IMT with Stochastic Error-Correction

A common problem in IMT arises when the user sets a prefix which cannot be explained by the statistical models. To solve this problem, IMT systems typically include ad-hoc error-correction techniques to guarantee that the suffixes can be generated (Barrachina et al., 2009). As an alternative to this heuristic approach, Ortiz-Martínez (2011) proposed a formalization of the IMT framework that does include stochastic error-correction models in its statistical formalization. The starting point of this alternative IMT formalization accounts for the problem of finding the translation \( t \) that, at the same time, better explains the source sentence \( s \) and the prefix given by the user \( p \):

\[
\hat{t} = \arg \max_t \Pr(t | s, p)
\]

\[
= \arg \max_t \Pr(t) \cdot \Pr(s, p | t)
\]

(7)

(8)

The following naïve Bayes’ assumption is now made: the source sentence \( s \) and the user prefix \( p \) are statistically independent variables given the translation \( t \), obtaining:

\[
\hat{t} = \arg \max_t \Pr(t) \cdot \Pr(s | t) \cdot \Pr(p | t)
\]

(9)

where \( \Pr(t) \) can be approximated with a language model, \( \Pr(s | t) \) can be approximated with a translation model, and \( \Pr(p | t) \) can be approximated by an error correction model that measures the compatibility between the user-defined prefix \( p \) and the hypothesized translation \( t \).

Note that the translation result, \( \hat{t} \), given by Equation (9) may not contain \( p \) as prefix because every translation is compatible with \( p \) with a certain probability. Thus, despite being close, Equation (9) is not equivalent to the IMT formalization in Equation (6).

To solve this problem, we define an alignment, \( a \), between the user-defined prefix \( p \) and the hypothesized translation \( t \), so that the unaligned words of \( t \), in an appropriate order, constitute the suffix searched in IMT. This allows us to rewrite the error correction probability as follows:

\[
\Pr(p | t) = \sum_a \Pr(p, a | t)
\]

(10)

To simplify things, we assume that \( p \) is monotonically aligned to \( t \), leaving the potential word-reordering to the language and translation models.
Under this assumption, \( a \) determines an alignment for \( t \), such that \( t = t_p t_s \), where \( t_p \) is fully-aligned to \( p \) and \( t_s \) remains unaligned. Taking all these things into consideration, and following a maximum approximation, we finally arrive to the expression:

\[
(\hat{t}, \hat{a}) = \arg \max_{t, a} \Pr(t) \cdot \Pr(s \mid t) \cdot \Pr(p, a \mid t) \tag{11}
\]

where the suffix required in IMT is obtained as the portion of \( t \) that is not aligned with the user prefix.

In practice, we combine the models in Equation (11) in a log-linear fashion as it is typically done in SMT (see Equation (3)).

### 2.4 Alternative Formalization for IMT with Stochastic Error-Correction

Alternatively to Equation (11), we can operate from Equation (9) and reach a different formalization for IMT with error-correction. We can re-write the first and last terms of Equation (9) as:

\[
\Pr(t) \cdot \Pr(p \mid t) = \Pr(p) \cdot \Pr(t \mid p) \tag{12}
\]

As in the previous section, we introduce an alignment variable, \( a \), between \( t \) and \( p \), giving:

\[
\Pr(t \mid p) = \sum_a \Pr(t, a \mid p) \tag{13}
\]

\[
= \sum_a \Pr(a \mid p) \cdot \Pr(t \mid p, a) \tag{14}
\]

If we consider monotonic alignments, \( a \) defines again an alignment between a prefix of the system translation \( t_p \) and the user prefix, producing the suffix required in IMT \( t_s \) as the unaligned part. Thus, we can re-write \( \Pr(t \mid p, a) \) as:

\[
\Pr(t \mid p, a) = \Pr(t_p, t_s \mid p, a) \tag{15}
\]

\[
\approx \Pr(t_p \mid p, a) \cdot \Pr(t_s \mid p, a) \tag{16}
\]

where Equation (16) has been obtained following a naive Bayes’ decomposition.

Combining equations (12), (14), and (16) into Equation (9), and following a maximum approximation for the summation of the alignment variable \( a \), we arrive to the following expression:

\[
(\hat{t}, \hat{a}) = \arg \max_{t, a} \Pr(s \mid t) \cdot \Pr(t_p \mid p, a) \cdot \Pr(t_s \mid p, a) \tag{17}
\]

where \( \Pr(p) \) and \( \Pr(a \mid p) \) have been dropped down because the former does not participate in the maximization and the latter is assumed uniform.

The terms in this equation can be interpreted similarly as those in Equation (9): \( \Pr(s \mid t) \) is the translation model, \( \Pr(t_p \mid p, a) \) is the error-correction probability that measures the compatibility between the prefix \( t_p \) of the hypothesized translation and the user-defined prefix \( p \), and \( \Pr(t_s \mid p, a) \) is the language model for the corresponding suffix \( t_s \) conditioned by the user-defined prefix. Again, in the experiments we combine the different models in a log-linear fashion.

The main difference between the two alternative IMT formalization (Equations (11) and (17)) is that in the latter the suffix to be returned is conditioned by the user-validated prefix \( p \). Thus, in the following we will refer to Equation (11) as independent suffix formalization while we will denote Equation (17) by conditioned suffix formalization.

### 3 Statistical Models

We now present the statistical models used to estimate the probability distributions described in the previous section. Section 3.1 describes the error-correction model, while Section 3.2 describes the models for the conditional translation probability.

#### 3.1 Statistical Error-Correction Model

Following the vast majority of IMT systems described in the literature, we implement an error-correction model based on the concept of edit distance (Levenshtein, 1966). Typically, IMT systems use non-probabilistic error correction models. The first stochastic error correction model for IMT was proposed in (Ortiz-Martínez, 2011) and it is based on probabilistic finite state machines. Here, we propose a simpler approach which can be seen as a particular case of the previous one. Specifically, the proposed approach models the edit distance as a Bernoulli process where each character of the candidate string has a probability \( p_e \) of being erroneous. Under this interpretation, the number of characters that need to be edited \( E \) in a sentence of length \( n \) is a random variable that follows a binomial distribution, \( E \sim B(n, p_e) \), with parameters \( n \) and \( p_e \). The probability of performing exactly \( k \) edits in a
sentence of \( n \) characters is given by the following probability mass function:

\[
f(k; n, p_e) = \frac{n!}{k!(n-k)!} p_e^k (1 - p_e)^{n-k}
\]  

(18)

Note that this error-correction model penalizes equally all edit operations. Alternatively, we can model the distance with a multinomial distribution and assign different probabilities to different types of edit operations. Nevertheless, we adhere to the binomial approximation due to its simplicity.

Finally, we compute the error-correction probability between two strings from the total number of edits required to transform the candidate translation into the reference translation. Specifically, we define the error-correction distribution in Equation (11) as:

\[
\Pr(p \mid a) \approx \frac{|p|!}{k!(|p| - k)!} p_e^k (1 - p_e)^{|p| - k}
\]  

(19)

where \( k = \text{Lev}(p, t_a) \) is the character-level Levenshtein distance between the user-defined prefix \( p \) and the prefix \( t_a \) of the hypothesized translation \( t \) defined by alignment \( a \). The error-correction probability \( \Pr(t_p \mid p, a) \) in Equation (17) is computed analogously.

The probability of edition \( p_e \) is the single free parameter of this formulation. We will use a separate development corpus to find an adequate value for it.

### 3.2 Statistical Machine Translation Models

Next sections briefly describe the statistical translation models used to estimate the conditional probability distribution \( \Pr(s \mid t) \). A detailed description of each model can be found in the provided citations.

#### 3.2.1 Phrase-Based Translation Models

Phrase-based translation models (Koehn et al., 2003) are an instance of the noisy-channel approach in Equation (2). The translation of a source sentence \( s \) is obtained through a generative process composed of three steps: first, the \( s \) is divided into \( K \) segments (phrases), next, each source phrase, \( \tilde{s} \), is translated into a target phrase \( \tilde{t} \), and finally the target phrases are reordered to compose the final translation.

The usual phrase-based implementation of the translation probability takes a log-linear form:

\[
\Pr(s \mid t) \approx \lambda_1 \cdot |t| + \lambda_2 \cdot K + \\
\sum_{k=1}^{K} \left[ \lambda_3 \cdot \log(P(\tilde{s}_k \mid \tilde{t}_k)) + \lambda_4 \cdot d(j) \right]
\]  

(20)

where \( P(\tilde{s} \mid \tilde{t}) \) is the translation probability between source phrase \( \tilde{s} \) and target phrase \( \tilde{t} \), and \( d(j) \) is a function (distortion model) that returns the score of translating the \( k \)-th source phrase given that it is separated \( j \) words from the \((k-1)\)-th phrase. Weights \( \lambda_1 \) and \( \lambda_2 \) play a special role since they are used to control the number of words and the number of phrases of the target sentence to be generated, respectively.

#### 3.2.2 Hierarchical Translation Models

Phrase-based models have shown a very strong performance when translating between languages that have similar word orders. However, they are not able to adequately capture the complex relationships that exist between the word orders of languages of different families such as English and Chinese. Hierarchical translation models provide a solution to this challenge by allowing gaps in the phrases (Chiang, 2005):

\[
\text{yu X}_1 \text{you X}_2 \rightarrow \text{have X}_2 \text{with X}_1
\]

where subscripts denote placeholders for sub-phrases. Since these rules generalize over possible phrases, they act as discontinuous phrase pairs and may also act as phrase-reordering rules. Hence, they are not only considerably more powerful than conventional phrase pairs, but they also integrate reordering information into a consistent framework.

These hierarchical phrase pairs are formalized as rewrite rules of a synchronous context-free grammar (CFG) (Aho and Ullman, 1969):

\[
X \rightarrow< \gamma, \alpha, \sim>
\]  

(21)

where \( X \) is a non-terminal, \( \gamma \) and \( \alpha \) are both strings of terminals (words) and non-terminals , and \( \sim \) is a one-to-one correspondence between non-terminal occurrences in \( \gamma \) and \( \alpha \). Given the example above, \( \gamma \equiv \text{"yu X}_1 \text{you X}_2" \), \( \alpha \equiv \text{"have X}_2 \text{with X}_1" \), and \( \sim \) is indicated by the subscript numbers.

Additionally, two glue rules are also defined:

\[
S \rightarrow <S_1 X_2 , S_1 X_2 > \quad S \rightarrow <X_1 , X_1 >
\]
These give the model the option to build only partial translations using hierarchical phrases, and then combine them serially as in a phrase-based model.

The typical implementation of the hierarchical translation model also takes the form of a log-linear model. Let $s_δ$ and $t_δ$ be the source and target strings generated by a derivation $δ$ of the grammar. Then, the conditional translation probability is given by:

$$
\Pr(s_δ \mid t_δ) \approx \lambda_1 \cdot |t_δ| + \lambda_2 \cdot |δ| + \lambda_3 \cdot \#_g(δ) + \sum_{r \in δ} [λ_4 \cdot w(r)]
$$

where $|δ|$ denotes the total number of rules used in $δ$, $\#_g(δ)$ returns the number of applications of the glue rules, $r \in δ$ are the rules in $δ$, and $w(r)$ is the weight of rule $r$. Weights $λ_1$ and $λ_2$ have a similar interpretation as for phrase-based models, they respectively give some control over the total number of words and rules that conform the translation. Additionally, $λ_3$ controls the model’s preference for hierarchical phrases over serial combination of phrases. Note that no distortion model is included in the previous equation. Here, reordering is defined at rule level by the one-to-one non-terminal correspondence. In other words, reordering is a property inherent to each rule and it is the individual score of each rule what defines, at each step of the derivation, the importance of reordering.

It should be noted that the IMT formalizations presented in Section 2 can be applied to other hierarchical or syntax-based SMT models such as those described in (Zollmann and Venugopal, 2006; Shen et al., 2010).

4 Search

In offline MT, the generation of the best translation for a given source sentence is carried out by incrementally generating the target sentence\(^2\). This process fits nicely into a dynamic programming (DP) (Bellman, 1957) framework, as hypotheses which are indistinguishable by the models can be recombined. Since the DP search space grows exponentially with the size of the input, standard DP search is prohibitive, and search algorithms usually resort to a beam-search heuristic (Jelinek, 1997).

Due to the demanding temporal constraints inherent to any interactive environment, performing a full search each time the user validates a new prefix is unfeasible. The usual approach is to rely on a certain representation of the search space that includes the most probable translations of the source sentence. The computational cost of this approach is much lower, as the whole search for the translation must be carried out only once, and the generated representation can be reused for further completion requests.

Next, we introduce hypergraphs, the formalism chosen to represent the search space of both phrase-based and hierarchical systems (Section 4.1). Then, we describe the algorithms implemented to search for suffix completions in them (Section 4.2).

4.1 Hypergraphs

A hypergraph is a generalization of the concept of graph where the edges (now called hyperedges) may connect several nodes (hypernodes) at the same time. Formally, a hypergraph is a weighted acyclic graph represented by a pair $⟨V, E⟩$, where $V$ is a set of hypernodes and $E$ is a set of hyperedges. Each hyperedge $e ∈ E$ connects a head hypernode and a set of tail hypernodes. The number of tail nodes is called the arity of the hyperedge and the arity of a hypergraph is the maximum arity of its hyperedges.

We can use hypergraphs to represent the derivations for a given CFG. Each hypernode represents a partial translation generated during the decoding process. Each ingoing hyperedge represents the rule with which the corresponding non-terminal was substituted. Moreover, hypergraphs can represent a whole set of possible translations. An example is

\(^2\)Phrase-based systems follow a left-to-right generation order while hierarchical systems rely on a CYK-like order.
shown in Figure 2. Two alternative translations are constructed from the leaf nodes (1, 2 and 3) up to the root node (6) of the hypergraph. Additionally, hypernodes and hyperedges may be shared among different derivations if they represent the same information. Thus, we can achieve a compact representation of the translation space that allows us to derive efficient search algorithms.

Note that word-graphs (Ueffing et al., 2002), which are used to represent the search space for phrase-based models, are a special case of hypergraphs in which the maximum arity is one. Thus, hypergraphs allow us to represent both phrase-based and hierarchical systems in a unified framework.

### 4.2 Suffix Search on Hypergraphs

Now, we describe a unified search process to obtain the suffix $t_s$ that completes a prefix $p$ given by the user according to the two IMT formulations (Equation (11) and Equation (17)) described in Section 2.

Given an hypergraph, certain hypernodes define a possible solution to the maximization defined in the two IMT formulations. Specifically, only those hypernodes that generate a prefix of a potential translation are to be taken into account. The probability of the solution defined by each hypernode has two components, namely the probability of the SMT model (given by the language and translation models) and the probability of the error-correction model. On the one hand, the SMT model probability is given by the translation of maximum probability through the hypernode. On the other hand, the error-correction probability is computed between $p$ and the partial translation of maximum probability actually covered by the hypernode. Among all the solutions defined by the hypernodes, we finally select that of maximum probability.

Once the best-scoring hypernode is identified, the rest of the translation not covered by it is returned as the suffix completion required in IMT.

### 5 Experimental Framework

The models and search procedure introduced in the previous sections were assessed through a series of IMT experiments with different corpora. These corpora, the experimental methodology, and the evaluation measures are presented in this section.

#### 5.1 EU and TED corpora

We tested the proposed methods in two different translation tasks each one involving a different language pair: Spanish-to-English (Es–En) for the EU (Bulletin of the European Union) task, and Chinese-to-English (Zh–En) for the TED (TED4 talks) task.

The EU corpora were 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. Particularly, the chosen Es–En corpus was part of the evaluation of the TransType2 project (Barrachina et al., 2009). The TED talks is a collection of recordings of public speeches covering a variety of topics, and for which high quality transcriptions and translations into several languages are available. The Zh–En corpus used in the experiments was part of the MT track in the 2011 evaluation campaign of the workshop on spoken language translation (Federico et al., 2011). Specifically, we used the dev2010 partition for development and the test2010 partition for test.

We process the Spanish and English parts of the EU corpus to separate words and punctuation marks keeping sentences truecase. Regarding the TED corpus, we tokenized and lowercased the English part (Chinese has no case information), and split Chinese sentences into words with the Stanford word

---

<table>
<thead>
<tr>
<th></th>
<th>EU (Es/En)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Development</td>
<td>Test</td>
</tr>
<tr>
<td>Sentences</td>
<td>214K</td>
<td>400</td>
<td>800</td>
</tr>
<tr>
<td>Token</td>
<td>5.9M / 5.2M</td>
<td>12K / 10K</td>
<td>23K / 20K</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>97K / 84K</td>
<td>3K / 3K</td>
<td>5K / 4K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>TED (Zh/En)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Development</td>
<td>Test</td>
</tr>
<tr>
<td>Sentences</td>
<td>107K</td>
<td>934</td>
<td>1664</td>
</tr>
<tr>
<td>Token</td>
<td>2M / 2M</td>
<td>22K / 20K</td>
<td>33K / 32K</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>42K / 52K</td>
<td>4K / 3K</td>
<td>4K / 4K</td>
</tr>
</tbody>
</table>

Table 1: Main figures of the processed EU and TED corpora. K and M stand for thousands and millions of elements respectively.
segmenter (Tseng et al., 2005). Table 1 shows the main figures of the processed EU and TED corpora.

5.2 Model Estimation and User Simulation

We used the standard configuration of the Moses toolkit (Koehn et al., 2007) to estimate one phrase-based and one hierarchical model for each corpus; log-linear weights were optimized by minimum error-rate training (Och, 2003) with the development partitions. Then, the optimized models were used to generate the word-graphs and hypergraphs with the translations of the development and test partitions.

A direct evaluation of the proposed IMT procedures involving human users would have been slow and expensive. Thus, following previous works in the literature (Barrachina et al., 2009; González-Rubio et al., 2010), we used the references in the corpora to simulate the translations that a human user would want to obtain. Each time the system suggested a new translation, it was compared to the reference and the longest common prefix (LCP) was obtained. Then, the first non-matching character was replaced by the corresponding character in the reference and a new system suggestion was produced. This process is iterated until a full match with the reference was obtained.

Finally, we used this user simulation to optimize the value for the probability of edition \( p_e \) in the error-correction model (Section 3.1), and for the log-linear weights in the proposed IMT formulations. In this case, these values were chosen so that they minimize the estimated user effort required to interactively translate the development partitions.

5.3 Evaluation Measures

Different measures have been adopted to evaluate the proposed IMT approach. On the one hand, different IMT systems can be compared according to the effort needed by a human user to generate the desired translations. This effort is usually estimated as the number of actions performed by the user while interacting with the system. In the user simulation described above these actions are: looking for the next error and moving the mouse pointer to that position (LCP computation), and correcting errors with some key strokes. Hence, we implement the following IMT effort measure (Barrachina et al., 2009):

**Key-stroke and mouse-action ratio (KSMR):** number of key strokes plus mouse movements performed by the user, divided by the total number of characters in the reference.

On the other hand, we also want to compare the proposed IMT approach against a conventional CAT approach without interactivity, such as a decoupled post-edition system. For such systems, character-level user effort is usually measured by the **Character Error Rate** (CER). However, it is clear that CER is at a disadvantage due to the auto-completion approach of IMT. To perform a fairer comparison between post-edition and IMT, we implement a post-editing system with autocompletion. Here, when the user enters a character to correct some incorrect word, the system automatically completes the word with the most probable word in the task vocabulary. To evaluate the effort of a user using such a system, we implement the following measure proposed in (Romero et al., 2010):

**Post-editing key stroke ratio (PKSR):** using a post-editing system with word-autocompleting, number of user key strokes divided by the total number of reference characters.

The counterpart of PKSR in an IMT scenario is (Barrachina et al., 2009):

**Key-stroke ratio (KSR):** number of key strokes, divided by the number of reference characters.

PKSR and KSR are fairly comparable and the relative difference between them gives us a good estimate of the reduction in human effort that can be achieved by using IMT instead of a conventional post-edition system.

We also evaluate the quality of the automatic translations generated by the MT models with the widespread BLEU score (Papineni et al., 2002).

Finally, we provide both confidence intervals for the results and statistical significance of the observed differences in performance. Confidence intervals were computed by pair-wise re-sampling as in (Zhang and Vogel, 2004) while statistical significance was computed using the Tukey’s HSD (honest significance difference) test (Hsu, 1996).
Table 2: BLEU score of the word-graphs (WG) and hypergraphs (HG) used to implement the IMT procedures.

<table>
<thead>
<tr>
<th></th>
<th>EU WG</th>
<th>EU HG</th>
<th>TED WG</th>
<th>TED HG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-best BLEU [%]</td>
<td>45.0</td>
<td>45.1</td>
<td>11.0</td>
<td>11.2</td>
</tr>
<tr>
<td>1000-best Avg. BLEU [%]</td>
<td>43.6</td>
<td>44.2</td>
<td>10.2</td>
<td>11.0</td>
</tr>
</tbody>
</table>

Table 3: IMT results (KSMR [%]) for the EU and TED tasks using the independent suffix formalization (ISF) and the conditioned suffix formalization (CSF) using both phrase-based (PB) and hierarchical (HT) translation models. For each task, the best result is displayed boldface, an asterisk * denotes a statistically significant better result (99% confidence) with respect to ISF with PB, and a star ⋆ denotes a statistically significant difference with respect to all the other systems.

<table>
<thead>
<tr>
<th>IMT Setup</th>
<th>EU</th>
<th>TED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PB</td>
<td>HT</td>
</tr>
<tr>
<td>ISF</td>
<td>27.4±.5</td>
<td>26.5±.5*</td>
</tr>
<tr>
<td>CSF</td>
<td>26.6±.5*</td>
<td>25.1±.5*</td>
</tr>
</tbody>
</table>

Table 4: Estimation of the effort required to translate the test partition of the EU and TED tasks using post-editing with word-completion (PE) and IMT under the independent suffix formalization (IMT). We used hierarchical MT in both approaches. In parenthesis we display the estimated effort reduction of IMT with respect to PE.

<table>
<thead>
<tr>
<th></th>
<th>EU PE</th>
<th>IMT PE</th>
<th>TED PE</th>
<th>IMT PE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKSR [%]</td>
<td>27.1</td>
<td>14.1 (48%)</td>
<td>40.8</td>
<td>29.7 (27.2%)</td>
</tr>
<tr>
<td>KSR [%]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6 Results

We start by reporting conventional MT quality results to test if the generated word-graphs and hypergraphs encode translations of similar quality. Table 2 displays the quality (BLEU (Papineni et al., 2002)) of the automatic translations generated for the test partitions. The lower 1-best BLEU results obtained for TED show that this is a much more difficult task than EU. Additionally, the similar average BLEU results obtained for the 1000-best translations indicate that word-graphs and hypergraphs encode translations of similar quality. Thus, the IMT systems that use these word-graphs and hypergraphs can be compared in a fair way.

Then, we evaluated different setups of the proposed IMT approach. Table 3 displays the IMT results obtained for the EU and TED tasks. We report KSMR (as a percentage) for the independent suffix formalization (ISF) and the conditioned suffix formalization (CSF) using both phase-based (PB) and hierarchical (HT) translation models. The KSMR result of ISF using a phrase-based model can be considered our baseline since this setup is comparable to that used in (Barrachina et al., 2009). Results for HT consistently outperformed the corresponding results for PB. Similarly, results for CSF were consistently better than those for ISF. More specifically, no statistically significant difference were found between ISF with HT and CSF with PB but both statistically outperformed the baseline (ISF with PB). Finally, CSF with HT statistically outperformed the other three configurations reducing KSMR by ~2.2 points with respect to the baseline. We hypothesize that the better results of HT can be explained by its more efficient representation of word reordering. Regarding the CSF, its better results are due to the better suffixes that can be obtained by taking into account the actual prefix validated by the user.

Finally, we compared the estimated human effort required to translate the test partitions of the EU and TED corpora with the best IMT configuration (independent suffix formalization with hierarchical translation model) and a conventional post-editing (PE) CAT system with word-completion. That is, when the user corrects a character, the PE system automatically proposes a different word that begins with the given word prefix but, obviously, the rest of the sentence is not changed. According to the results, the estimated human effort to generate the error-free translations was significantly reduced with respect to using the conventional PE approach. IMT can save about 48% of the overall estimated effort for the EU task and about 27% for the TED task.

7 Summary and Future Work

We have proposed a new IMT approach that uses hierarchical SMT models as its underlying translation technology. This approach is based on a statistical formalization previously described in the literature that includes stochastic error correction. Additionally, we have proposed a refined formalization that improves the quality of the IMT suffixes by taking
into account the prefix validated by the user. Moreover, since word-graphs constitute a particular case of hypergraphs, we are able to manage both phrase-based and hierarchical translation models in a unified IMT framework.

Simulated results on two different translation tasks showed that hierarchical translation models outperform phrase-based models in our IMT framework. Additionally, the proposed alternative IMT formalization also allows to improve the results of the IMT formalization previously described in the literature. Finally, the proposed IMT system with hierarchical SMT models largely reduces the estimated user effort required to generate correct translations in comparison to that of a conventional post-edition system. We look forward to corroborating these result in test with human translators.

There are many ways to build on the work described here. In the near future, we plan to explore the following research directions:

- Alternative IMT scenarios where the user is not bounded to correct translation errors in a left-to-right fashion. In such scenarios, the user will be allowed to correct errors at any position in the translation while the IMT system will be required to derive translations compatible with these isolated corrections.

- Adaptive translation engines that take advantage of the user’s corrections to improve its statistical models. As the translator works and corrects the proposed translations, the translation engine will be able to make better predictions. One of the first works on this topic was proposed in (Nepveu et al., 2004). More recently, Ortiz-Martínez et al. (2010) described a set of techniques to obtain an incrementally updatable IMT system, solving technical problems encountered in previous works.

- More sophisticated measures to estimate the human effort. Specifically, measures that estimate the cognitive load involve in reading, understanding and detecting an error in a translation (Foster et al., 2002), in contrast KSMR simply considers a constant cost. This will lead to a more accurate estimation of the improvements that may be expected by a human user.

Acknowledgments

Work supported by the European Union 7th Framework Program (FP7/2007-2013) under the CasMaCat project (grant agreement n° 287576), by Spanish MICINN under grant TIN2012-31723, and by the Generalitat Valenciana under grant ALMPR (Prometeo/2009/014).

References


Attachment E

Task 2.5

Philipp Koehn.

Efficient prediction for interactive machine translation for syntax-based models.

Not yet published, 2013.
Efficient Prediction for Interactive Machine Translation
for Syntax-Based Models

Philipp Koehn
School of Informatics
University of Edinburgh
Scotland, United Kingdom
pkoehn@inf.ed.ac.uk

Abstract

We propose two exact dynamic programming algorithms to solve the approximate prefix matching problem for search forests generated by syntax-based translation models. We also propose three refinements and assess the impact of the methods on processing speed on the CASMACAT field trial data.

The contemporary increase in machine translation quality enables professional translators to work faster by post-editing machine translation output than by translating from scratch. This has been also demonstrated in the first CASMACAT field trial.

Interactive machine translation (WP2) aims to provide a more user-oriented editing environment for human translators to take advantage of machine translation. Its functionality is similar to the auto-complete functionality for text input in mobile devices that reduce typing by proposing word completions and next words. Informed by the translation model, interactive machine translation proposes a sentence completion, optimized to the user’s partial translation at any point in time.

So far, interactive machine translation has been applied to word-based and phrase-based models. In this paper, we extend the approach to syntax-based models. For many language pairs, including Chinese–English or German–English, such models have shown superior translation quality. We would like to leverage these gains for interactive machine translation.

We propose a top-down and a bottom-up search algorithm which give exact solutions to the approximate matching problem. We also propose a number of refinements. The methods are evaluated on the first field trial data of the CASMACAT project.

1 Interactive Machine Translation

Interactive machine translation allows the translator to write from left to right and proposes a sentence completion at any point. Initially, the sentence completion suggestion is the best translation according to the machine translation model, but if the translator deviates from the suggested path, the interactive machine translations updates its suggestion.

Sentence completion prediction typically operates on the stored search space of the machine translation decoder. The task requires matching the partial user translation (called the prefix) against paths in the search graph. Since the user may type in word sequence that do not occur in the search graph, approximate matching is required. The primary goal is to find paths with minimal string edit distance, and then choose the most probably path among these.

2 Syntax-Based Models

Current syntax-based statistical machine translation approaches model bilingual sentence pairs with synchronous context-free grammars (SCFG). These grammars may have linguistically motivated non-terminals on the source side, or target side, or both. But they may also just employ a generic non-terminal $X$ in a version of the approach that is commonly called hierarchical phrase-based model.

Target-side SCFG rules may translate words and phrases

$$\text{NP} \rightarrow \text{die Katze} ; \text{the cat}$$
$$\text{VBZ} \rightarrow \text{jagt} ; \text{chases}$$

build up syntactic structure (indices at the non-terminals indicate alignment)

$$S \rightarrow X_1 X_2 ; \text{NP}_1 \text{VP}_2$$
$$\text{NP} \rightarrow X_1 X_2 X_3 ; \text{DET}_1 \text{NN}_3 \text{JJ}_2$$
or mix words and non-terminals
Figure 1: Parse forest generated by a SCFG translation model search process for the German input sentence die Maus jagt die Katze.

\[
\text{NP} \rightarrow \text{die } X_1; \text{the NP}_1 \\
\text{NP} \rightarrow \text{kann } X_1 \text{ machen; can make NP}_1
\]

Translating with SCFG requires building up the parse structure and translations for parts of the source sentence by bottom-up chart decoding. The end result is a parse forest that contains a compact representation of many parse trees considered during search.

See Figure 1 for an example parse forest, generated by the search for translations of the German sentence die Maus jagt die Katze. At the bottom of each parse tree are lexical translations (e.g., Maus into mouse, or die Katze into the cat), which are then used as building blocks for constituents covering larger spans of the input sentence. For instance the rule VP \( \rightarrow \text{jagt } X_1; \text{chases NP}_1 \) builds on the already generated constituent the cat.

Each constituent in the parse forest is scored according to several feature functions, such as the rule probability or language model score, indicated in Figure 1 with \( m:\text{score} \). The forest is pruned during search by discarding low-scoring constituents.

When two constituents covering the same span of source words are identical from the view of future rule applications (because they have the same constituent label and language model context), the worse one can be discarded (if the goal is to find the best scoring translation) or recombined (if we want to preserve a parse forest containing many possible derivations).

The goal of SCFG decoding is to find the highest scoring constituent that covers the entire source sentence. As a by-product, a parse forest is generated that we utilize for interactive machine translation.

3 Top-Down Search

Given the parse forest generated by the SCFG search and the user prefix (the partial translation already entered by the user), the interactive machine translation problem is to find the most probably completion of the sentence based on a match of the prefix against the parse forest with the minimal number of edits.

The first algorithm we propose solves the problem of matching prefix against parse forest with top-down dynamic programming search. The motivation for top-down search is that the user input has to be matched only partially, and parts of the search forest that cover the rest of the sentence do not have to be considered in the matching process. By following the parse tree top down, matching the user prefix left-to-right, we can restrict the search to the relevant part of the parse forest.

3.1 Example

See Figure 2 for an illustration of this process, based on the user prefix the small cat chases, and the maximum edit cost of 1 (more on the maximum edit cost later). We start at the sentence node (1), which was generated with the rule S \( \rightarrow X_1 X_2; \text{NP}_1 \text{ VP}_2 \). We following the first non-terminal of the rule’s right hand side, NP. This directs us to Node (7), which was generated by the fully lexical rule NP \( \rightarrow \text{die Katze; the cat}. \)

When encountering rules with lexical items, we...
have to match the user prefix against the words in the rule. We need to match all words in the rule, unless we reach the end of the user prefix or when additional words of the user prefix can only be matched with deletions of words from the prefix.

In this case, the following three ways of matching the rule and the prefix are computed:

<table>
<thead>
<tr>
<th>prefix</th>
<th>edit sequence</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>match, insertion</td>
<td>1</td>
</tr>
<tr>
<td>the small</td>
<td>match, substitution</td>
<td>1</td>
</tr>
<tr>
<td>the small cat</td>
<td>match, deletion, match</td>
<td>1</td>
</tr>
</tbody>
</table>

Here, the dynamic programming aspect of the search comes in play. Once, computed, the table is stored with node (7). It contains the optimal matches when starting with the first user prefix word the. If we visit the node again at that word position, we do not need to compute the table again, but can use the cached table.

After complete processing of node (7), we return to node (1). We now process the next non-terminal VP, which points us to node (3), a result of the rule application NP → jagt X₁ ∶ chases NP₁. Since we only occurred an edit cost of 1, we are allowed no more edits. The only way to match the rule (partially) is:

<table>
<thead>
<tr>
<th>prefix</th>
<th>edit sequence</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>chases</td>
<td>match</td>
<td>0</td>
</tr>
</tbody>
</table>

This consumes the entire user prefix, and hence we find a solution with edit cost 1. The predicted sentence completion, by following the NP in node (3), is the mouse.

This completes also the processing of node (3) and node (1). Next, we explore the other sentence node, such as node (2) in the example, until we are certain that no better match can be found.

3.2 Algorithm

See Algorithm 1 for the pseudo-code of the algorithm. The matching routine process-node is called with an increasing number of allowable edits (error), until matches are found. Among the matches, the one with the highest score is selected.

The function process-node loops through all edges that lead to this node, and through each symbol on the right hand side of the rule that was used to create the edge. Before a symbol is matched, any number of deletions may take place (lines 8–14). If a symbol is a nonterminal, it may be either matched (lines 19–20), substituted (line 23) or deleted.

### Algorithm 1 Top-down prefix-matching

**Input:** user prefix u, search forest f

**Output:** best path p

1: allowable error e = 0
2: while Zfinal == {} and error e < length(p) do
3:   for all node ∈ end-node-set do
4:     Z = process-node(node,0,e)
5:     Zfinal = {z ∈ Z : word(z) == length(u)}
6:   end for
7: end while
8: return best scoring z ∈ Zfinal

process-node(node, start-word-covered, max-error):

1: if cache for (node, start-word-covered, e ≤ max-error) exists then
2:   return cache
3: end if
4: for all edge e in node do
5:   add (state-word-covered,0, score(e), e) to set Z
6: for all symbol s in right-hand-side of edge e rule do
7:   Znext = Z
8: // deletions
9: for all states z ∈ Z do
10:   for i=1..min(max-error – error(z), length(u) – word(z)) do
11:     add (word(z)+i, error(z)+i, score(z), path(z) + “deletion” * i) to Znext
12:   end for
13: end for
14: Z = Znext
15: // consume a symbol from the rule
16: for all states z ∈ Z do
17:   if s is non-terminal then
18:     // word: match, substitute or delete
19:     if u(word(z)) == s then
20:       add (word(z)+1, error(z), score(z), path(z) + “match” to Znext
21:     else
22:       if error(z) < max-error then
23:         add (word(z)+1, error(z)+1, score(z), path(z) + “substitution”) to Znext
24:       add (word(z), error(z)+1, score(z), path(z) + “insertion”) to Znext
25:     end if
26: end if
27: else
28:     // non-terminal: recurse
29:     Zsub = process-node(node(s), word(z), max-error – error(z), path(z))
30:     for all z₂ ∈ Zsub do
31:       add z + z₂ to Znext
32:     end for
33: end if
34: end for
35: Z = {}
36: for all z ∈ Znext do
37:   if word(z) == length(u) then
38:     add z to Zfinal
39: else
40:   add z to Z
41: end if
42: end for
43: end for
44: end for
45: return Z + Zfinal
or inserted (line 24). When encountering a non-
terminal symbol, we recursively call PROCESS-
NODE and merge the resulting states (lines 29–32).

If after matching a symbol the entire prefix is
matches, we store the state in the set of final
states (line 38), otherwise we continue to the next sym-
bol. Finally, all final state and partially processed
states are returned.

Matching states have the fields:
• last word covered
• error (edit cost)
• score (model)
• path (edit sequence)

The cache contains the additional information
• node
• start word covered

Note that given a state’s path, the edit sequence
between user prefix and partial translation in the
parse forest can be traced back and the sentence
completion can be obtained.

3.3 Complexity
In the worst case, every node is visited. Although
each node may be processed multiple times, re-
sults are cached the first time, and hence no ad-
ditional processing is carried out. However, if the
node processing is called with different prefix start
word positions, or with a bigger allowable error
than before, processing does take place.

With a maximum error of the length of the
prefix, and starting positions ranging from 0
to one word before the end of the prefix, the
number of times PROCESS-NODE is called is
\(O(length(prefix)^2 \times count(node))\).

While processing a node, all its edges are con-
siders, so the maximum number of times edges are
processed is \(O(length(prefix)^2 \times count(edge))\).

During processing of edges, for each processed
symbol, a set of partial matching states is created
and processed. Note that in the sketch of the al-
thor's algorithm, this set could potentially become very
large. However, it can be reduced to the optimal
match for each matched position of the prefix. If
there are two ways to match the prefix to the exact
same position, then one has either a worse error
(edit cost), or worse model score and hence can
be safely discarded. The most expensive process-
ing step for a symbol is the merging in lines 30–32,
where \(O(length(prefix))\) partial states \(z\) have
to combined with \(O(length(prefix))\) sub-states \(z_2\).

(a) Match of Substring

(b) One Deletion

(c) Trailing Word

Figure 3: State information for bottom-up search

This gives an overall complexity of
\(O(length(prefix)^4 \times count(edge))\).

4 Bottom-Up Search

Our second algorithm matches the user prefix
against the parse forest bottom-up. The main idea
is that a leaf edge typically matches the prefix only
in one position, so only that information needs to
stored.

4.1 Example

Figure 3 illustrates how match information is
stored. In the simplest case, the edge string
matches a substring of the user prefix. We need
to record the start and final position of the match.
Example (a) shows a match of edge string \(cd\) to
user string \(cd\).

If the edge string only matches with edits, as in
the example, this edit error is recorded in error.
Example (b) has a deletion of a prefix word, so
the match between \(cde\) and \(ce\) has an string edit
distance of 1.

We record leading and trailing words that do not
match the user prefix differently, by explicitly storing the number of words before and after. In example (c), the last word $x$ of the edge string $\text{cex}$ does not match the user prefix, so we record this as $\text{before}=1$, and not as part of the edit error.

Mismatched leading and training words (before and after) recorded differently, since they affect how two neighboring edges can be combined. For instance, if the user prefix is $\text{abcde}$, and we merge the edges $\text{abx}$ and $\text{yde}$, then merging the match states ($\text{before}=0$, $\text{start}=1$, $\text{error}=0$, final=2, after=1) and (1,4,0,5,0) can consolidate the three mismatches (after$_1=1$, before$_2=1$, not covering word $c$) into just 2 edits (maximum of 2 additional words in combined edges, 1 additional word in user prefix).

4.2 Algorithm

Algorithm 2 displays the pseudo-code for the bottom-up search method.

The beginning of the algorithm is similar to the top-down search. The parse forest is matched against the user prefix with increasing number of allowed maximum edits (lines 2–7 in main function). The parse tree is also traversed top down (calling PROCESS-NODE-BOTTOM-UP from end nodes in line 4, recursive calls to child nodes in line 19 of PROCESS-EDGE-BOTTOM-UP). However, computation of matching states is done only if all child node states have been completely processed, hence it is a bottom up search.

Caching of results is done for nodes (lines 12–16 of PROCESS-NODE-BOTTOM-UP) and edges (lines 54–58 of PROCESS-EDGE-BOTTOM-UP). Bottom-up search allows for exhaustive matching of edges and nodes against the user prefix. However, for nodes higher up in the tree, which span over many possible strings, there may be many ways how the user prefix may be matched with unlimited error. So, if exhaustive processing becomes computationally prohibitive, then the matching process prunes out matches that have more than allowable error (lines 6–8 of PROCESS-EDGE-BOTTOM-UP).

Maximum error limited caching does imply that states that have been processed with a lower maximum error in earlier iterations of the main loop may have to be revisited. Given the unknown minimum error that leads to a valid match between parse forest and user prefix, it is not clear at run time how to best handle the trade-off between avoiding unnecessary exhaustive computations and avoiding revisiting nodes with a higher allowable maximum error.

4.3 Merging Partial States

The heart of the algorithm is the matching of partial states (lines 26–49 of PROCESS-EDGE-BOTTOM-UP). In the target production of a grammar rule $A \rightarrow \alpha_1..\alpha_n$, the symbols on the right hand side ($\alpha_1, ..., \alpha_n$) may be words or non-terminals. Both types are matched against the user prefix, resulting in a set of match states.

Matching words results either in one (or more) single word exact matches against the user prefix (line 13), or an error state (line 15). The match result sets for non-terminals are retrieved by a recursive call to PROCESS-NODE-BOTTOM-UP (line 19).

The $n$ result sets for the right-hand-side symbols $\alpha_1, ..., \alpha_n$, are merged in pairwise from left to right (loop starts at line 5). For any of the pairwise merging steps, a set of partial states ($Z_{\text{partial}}$) is merged with the result states from the most recent symbol ($Z_{\text{symbol}}$), resulting in a new set of (partial) states $Z$ (loops start at line 24).

When matching two states a number of distinct cases have to be distinguished:

- If both of the states are error states (i.e., their string do not match any of the user prefix words), then the resulting state is also an error state with the combined error count (lines 26–27).
- If one of the states is an error state, then its error (i.e., number of words in the string) has to be added either as before or after word count to the regular matching state (lines 28–31).
- If the two states attach seamlessly, then no additional error is incurred when they are merged into a new state (lines 32–33).
- If the two states otherwise match non-overlapping substrings of the user prefix in the right order, then the merging into a new state incurs various errors depending on the number of relevant before and after words of the states and the distance between the two matches in the user prefix (lines 24–38).
- If the two states match overlapping substrings of the user prefix, then it is not guaranteed
Algorithm 2 Bottom-up prefix-matching

Input: user prefix $u$, search forest $f$
Output: best path $p$

1: allowable error $e = 0$
2: while $Z_{\text{final}} = \emptyset$ and error $e < \text{length}(p)$ do
3: for all node $\in$ end-node-set do
4: $Z = \text{PROCESS-NODE-BOTTOM-UP}(\text{node}, e)$
5: $Z_{\text{final}} = \{ z \in Z : \text{word}(z) == \text{length}(u) \}$
6: end for
7: end while
8: return best scoring $z \in Z_{\text{final}}$

PROCESS-NODE-BOTTOM-UP(node, max-error):
1: if cache status for (node, max-error) complete or sufficient for max-error then
2: return cache
3: end if
4: pruning-flag = false
5: min-error = $\infty$
6: max-score = 0
7: for all edge $e$ in node do
8: $(p, z) = \text{PROCESS-EDGE-BOTTOM-UP}(\text{edge, max-error})$
9: if pruning-flag = true if $p == true$
10: add $z$ to result-set $Z$
11: end for
12: if pruning-flag == true then
13: add result-set to cache with max-error
14: else
15: add result-set to complete cache
16: end if
17: return (pruning-flag, $Z$)

PROCESS-EDGE-BOTTOM-UP(edge, max-error):
1: if cache status for (edge, max-error) complete or sufficient for max-error then
2: return cache
3: end if
4: $Z_{\text{partial}} = \{ (\text{error} = 0, \text{score} = \text{score}(\text{edge})) \}$
5: for all symbol $s$ in right-hand-side of edge $e$ rule do
6: if $Z_{\text{partial}}$ too large then
7: reduce $Z_{\text{partial}}$ to max-error
8: pruning-flag == true
9: end if
10: if $s$ is word then
11: // word: match against prefix
12: for all word position $i$ in prefix that matches word do
13: add state (start=$i$, before=0, error=0, after=0, final=i+1) to $Z_{\text{symbol}}$
14: end for
15: add state (error=1) to $Z_{\text{symbol}}$ if no word matches
16: else
17: // non-terminal: retrieve sub-states
18: min-error = $\min(\text{error}(Z_{\text{partial}}))$
19: $(p, Z_{\text{symbol}}) = \text{PROCESS-NODE-BOTTOM-UP}(\text{child-node}, \text{max-error} - \text{min-error})$
20: pruning-flag = true if $p == true$
21: end if
22: end if
23: $Z = \emptyset$
24: for all states $Z_{\text{partial}} \in Z_{\text{partial}}$ do
25: for all states $Z_{\text{symbol}} \in Z_{\text{symbol}}$ do
26: if both error states then
27: add state (error=error($Z_{\text{partial}}$)+error($Z_{\text{symbol}}$), score=score($Z_{\text{partial}}$)+score($Z_{\text{symbol}}$) to $Z$
28: else if $Z_{\text{partial}}$ error state then
29: add state based on $Z_{\text{symbol}}$, add error($Z_{\text{partial}}$) to $Z_{\text{symbol}}$, before, score($Z_{\text{partial}}$) to $Z$
30: else if $Z_{\text{symbol}}$ then
31: add state based on $Z_{\text{partial}}$, add error($Z_{\text{symbol}}$) to $Z_{\text{partial}}$, after, score($Z_{\text{symbol}}$) to $Z$
32: else if $Z_{\text{partial}}$ before $Z_{\text{symbol}}$ with gap then
33: surplus-prefix = start($Z_{\text{symbol}}$) - final($Z_{\text{partial}}$) + surplus($Z_{\text{partial}}$)
34: surplus-state = after($Z_{\text{partial}}$) + before($Z_{\text{symbol}}$)
35: gap-error = $\max($surplus-state, surplus-state$)$
36: add state (start=start($Z_{\text{partial}}$), before=before($Z_{\text{partial}}$), after=after($Z_{\text{partial}}$), error=error($Z_{\text{partial}}$)+error($Z_{\text{symbol}}$), final=final($Z_{\text{symbol}}$), score=score($Z_{\text{partial}}$)+score($Z_{\text{symbol}}$)) to $Z$
37: else if $Z_{\text{partial}}$ before $Z_{\text{symbol}}$ with overlap then
38: surplus-prefix = start($Z_{\text{symbol}}$) - final($Z_{\text{partial}}$) + surplus($Z_{\text{partial}}$)
39: surplus-state = after($Z_{\text{partial}}$) + before($Z_{\text{symbol}}$)
40: gap-error = $\max($surplus-state, surplus-state$)$
41: add state (start=start($Z_{\text{partial}}$), before=before($Z_{\text{partial}}$), after=after($Z_{\text{partial}}$), error=error($Z_{\text{partial}}$)+error($Z_{\text{symbol}}$)+gap-error, final=final($Z_{\text{symbol}}$), score=score($Z_{\text{partial}}$)+score($Z_{\text{symbol}}$)) to $Z$
42: else if $Z_{\text{partial}}$ overlaps or succeeds $Z_{\text{symbol}}$ then
43: matched-partial-words = final($Z_{\text{partial}}$)-start($Z_{\text{partial}}$)-before($Z_{\text{partial}}$)
44: matched-symbol-words = final($Z_{\text{symbol}}$)-start($Z_{\text{symbol}}$)-before($Z_{\text{symbol}}$)
45: if matched-partial-words + matched-symbol-words > start($Z_{\text{symbol}}$)-final($Z_{\text{partial}}$) then
46: overlap-error = final($Z_{\text{partial}}$)+start($Z_{\text{symbol}}$)
47: surplus-word-error = after($Z_{\text{partial}}$)+before($Z_{\text{symbol}}$)
48: add state (start=start($Z_{\text{partial}}$), before=before($Z_{\text{partial}}$), error=error($Z_{\text{partial}}$)+error($Z_{\text{symbol}}$)+overlap-error+surplus-word-error, after=after($Z_{\text{symbol}}$), final=final($Z_{\text{symbol}}$), score=score($Z_{\text{partial}}$)+score($Z_{\text{symbol}}$)) to $Z$
49: else
50: add states based on either state, treating the other as complete mismatch
51: end if
52: end if
53: end for
54: if pruning-flag == true then
55: add result-set to cache with max-error
56: else
57: add result-set to complete cache
58: end if
59: return (pruning-flag, $Z$)
that they can be merged with a lower error than considering one of the states as pure error state (condition in line 42). If they can be merged, then the error computation is somewhat complex (details in the code, lines 43–45).

### 4.4 Recombination of Matching States

A single string under a node in the parse forest may be matched in various ways against the user prefix. Table 1 gives an example for matching `xbxdx` against `abode`. In this example, we record the maximum match (matching both `b` and `d`, first line in the table) with one internal error. But we also record the matches of only one word, with no internal error. We may also record a pure error state, although this one can always be derived from the others.

### 4.5 Complexity

Opposed to the top-down search, the bottom-up algorithm requires the processing of each edge and node only once, if we complete an exhaustive processing of a node.

However, if many match states are generated for nodes, the combinatorial complexity of merging these states becomes computationally costly. For a rule with `n` symbols on the target right hand side, and `C` matching states per symbol, `C^n` match states results. While many of these states may be recombined, the polynomial cost of creating them remains to some degree.

### 5 Refinements

Independent of the choice of the top-down or bottom-up algorithm, a number of refinements that pre-process the search forest can be applied.

#### 5.1 Spurious Ambiguity

Most statistical machine translation systems allow for multiple derivations that yield the same surface string translations. This is also the case for SCFG grammar models, especially hierarchical phrase-based models.

To give a simple example, a model may have the rules

1. `X → die X ; the X`
2. `X → Katze ; cat`
3. `X → x Katze ; x cat`
4. `X → die ; the`
5. `X → die Katze ; the cat`

which allow for three different derivations (rules 1+2, rules 3+4, rule 5) that all result in the translation of `die Katze` into `the cat`. In the parse forest, these three edges will be connect into a single node.

Such spurious ambiguity can be detected straightforwardly. By computing the target strings for edges and nodes bottom up, we can find sets of edges with the same yield, and safely discard the lower-scoring ones. Note that this does not reduce the number of edges for each node to one, since state recombination during decoding is based on language model state equivalence, which is less strict than full string matching.

Removing irrelevant edges from the parse forest reduces processing time for out algorithms and can be done offline, so the cost of this processing should not be considered when measuring the efficiency of the online matching algorithms.

#### 5.2 Normalizing Non-Matching Words

Words in the parse forest that do not match any of the words in the user prefix can be normalized, i.e., all replaced with a junk token, since they all will have to treated as insertions or substitutions during matching. After doing so, we can carry out another round of spurious ambiguity reduction, as laid out in the previous section.

However, since this pre-processing step can only be carried out when the user prefix is known, its cost must be considered when assessing the search algorithms.

#### 5.3 Inside-Outside Pruning

The number of nodes in the search forest depends on the pruning settings of the decoder that produced it. Typically, in syntax based models, cube pruning is used, so a fixed number of edges are generated for each source language span.

However, especially in target-side syntax models, some source spans are unlikely to be covered

<table>
<thead>
<tr>
<th>before</th>
<th>start</th>
<th>error</th>
<th>final</th>
<th>after</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Canonic match states `xbxdx` against `abode`
by useful edges. To give an example, the span example, the in this sentence is unlikely to be translated into a syntactic constituent in a target language.

Some of the edges and nodes are removed, because they are not reachable from any end node that spans the entire input sentence. But we may want to additionally remove edges which can only be part of low-scoring derivations. For each edge, we can compute the inside-outside score, which is combination of the best score of any full derivation that contains this edge. The score is called inside-outside, because its efficient computation requires the keeping track of the optional score for each edge (inside score) and the score for the other edges involved in an optimal derivation (outside score).

Reducing the parse forest by discarding edges with bad inside-outside score, may lead to sentence completion predictions that have higher edit cost or worse parse scores (or both). However, given the need for real time processing, this may be a price worth paying.

6 Experiments

We explore the behavior of the algorithms and the refinements in an experimental setting which simulates user interaction.

6.1 Setup

We use the results of the casmacat field trial. In this trial, output (news stories) from an English–Spanish phrase-based Moses system was post-edited by professional translators. By translating the sentences again with an English–Spanish hierarchical phrase-based Moses system trained on the same data, we can achieve output that is similar to the starting point of the professional translator, thus we can expect that the parse forests may match it closely in many cases.

The goal now is to use the parse forests obtained by the hierarchical model to predict the next word in the user prefix, which are the prefixes of the translations created by the professional translator. This simulates the interaction of the professional translator with the interactive machine translation system.

Note that we are here less interested in the prediction accuracy, but in the speed of the algorithm. The prediction accuracy depends on many factors, such as the quality of the machine translation system, which are constant for all the variants of the algorithms that we explore here.

The implementation of the methods that is used in the experiments is written in Perl, which suffers from rather slow performance, but allowed for quicker implementations of the algorithms and refinements. By re-implementing the most successful methods in C++ we expect speed improvements by a factor of 10 to 100.

6.2 Influencing Factors

The run time of the algorithms depend on a number of factors, most crucially:

- length of the source sentence
- length of the user prefix
- number of edits

How these factors influence processing times is illustrated in Figure 4. The graph shows the progressing curves on how much time is needed to compute the prediction for a user prefix of increasing length for three sentences, using the top-down algorithm from Section 3.

The shortest sentence ends with an edit cost of 5 for the largest prefix (27 words), which takes 1.5 seconds to compute. In comparison, a 37-word sentence ends with final edit cost of 5 in 1.8 seconds, and the longest sentence of 47 words ends with 4 edits in 3.8 seconds.

What is clearly visible from the progressing graph are the jumps in processing time whenever the number of edits increases. Otherwise, the curves appear almost flat. The step increase is not independent from sentence length. Matching
<table>
<thead>
<tr>
<th>edits</th>
<th>ratio</th>
<th>top down</th>
<th>bottom up</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>33%</td>
<td>0.07s</td>
<td>0.26s</td>
</tr>
<tr>
<td>1</td>
<td>17%</td>
<td>0.14s</td>
<td>0.64s</td>
</tr>
<tr>
<td>2</td>
<td>16%</td>
<td>0.28s</td>
<td>2.26s</td>
</tr>
<tr>
<td>3</td>
<td>11%</td>
<td>0.65s</td>
<td>2.31s</td>
</tr>
<tr>
<td>4</td>
<td>11%</td>
<td>1.01s</td>
<td>3.31s</td>
</tr>
</tbody>
</table>

Table 2: Average processing time by edits for the core algorithms. The top-down algorithm is about 3–4 times faster than the bottom-up algorithm.

Table 3: Processing time reductions due to spurious ambiguity removal.

<table>
<thead>
<tr>
<th>edits</th>
<th>1</th>
<th>0.5</th>
<th>0.2</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>741 / 0.54s</td>
<td>650 / 0.07s</td>
<td>499 / 0.01s</td>
<td>416 / 0.01s</td>
</tr>
<tr>
<td>1</td>
<td>419 / 1.18s</td>
<td>225 / 0.14s</td>
<td>287 / 0.02s</td>
<td>258 / 0.01s</td>
</tr>
<tr>
<td>2</td>
<td>402 / 3.09s</td>
<td>309 / 0.28s</td>
<td>220 / 0.03s</td>
<td>205 / 0.01s</td>
</tr>
<tr>
<td>3</td>
<td>268 / 5.99s</td>
<td>217 / 0.65s</td>
<td>199 / 0.05s</td>
<td>226 / 0.02s</td>
</tr>
<tr>
<td>4</td>
<td>213 / 14.5s</td>
<td>225 / 1.01s</td>
<td>208 / 0.09s</td>
<td>188 / 0.03s</td>
</tr>
</tbody>
</table>

Table 3: Average processing time by edits for the core algorithms. The top-down algorithm is about 3–4 times faster than the bottom-up algorithm.

Table 4: Impact of inside-outside pruning. Discarding edges and nodes with a score difference bigger than a specified threshold (1, 0.5, 0.2, 0.1): Number of prefix matches and processing speed at different edit levels (top-down algorithm only).

<table>
<thead>
<tr>
<th>edits</th>
<th>1</th>
<th>0.5</th>
<th>0.2</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>741</td>
<td>650</td>
<td>499</td>
<td>416</td>
</tr>
<tr>
<td>1</td>
<td>419</td>
<td>225</td>
<td>287</td>
<td>258</td>
</tr>
<tr>
<td>2</td>
<td>402</td>
<td>309</td>
<td>220</td>
<td>205</td>
</tr>
<tr>
<td>3</td>
<td>268</td>
<td>217</td>
<td>199</td>
<td>226</td>
</tr>
<tr>
<td>4</td>
<td>213</td>
<td>225</td>
<td>208</td>
<td>188</td>
</tr>
</tbody>
</table>

Table 4: Impact of inside-outside pruning. Discarding edges and nodes with a score difference bigger than a specified threshold (1, 0.5, 0.2, 0.1): Number of prefix matches and processing speed at different edit levels (top-down algorithm only).

6.3 Comparison of Core Algorithms
Given the various influencing factors, we settled on reporting results separately based on how many edits are needed to match the user prefix against the parse forest.

Table 2 compares the two basic algorithms: the top-down and bottom up search. The top-down algorithm is about 3–4 times faster. This is independent of the number of edits needed, thus not giving evidence that either algorithms is better asymptotically.

Total processing time roughly doubles with each additional edit.

6.4 Reduction of Spurious Ambiguity
Our first refinement is the removal of spurious ambiguity from the parse forest. As explained in Section 5.1, multiple edges and nodes may yield exactly the same strings, so the weaker ones can be safely discarded. This is a pre-processing step that can be done offline, so it does not add to the run time of the online algorithm.

Table 3 shows that the time reduction is mostly around 20%–35%. What is encouraging is that we observe bigger reductions when more edits are needed. We see slightly bigger reductions with the top down algorithm.

6.5 Normalizing Non-Matching Words
By normalizing all words in the parse forest that do not match any of the input words, more aggressive spurious ambiguity removal can be carried out (Section 5.2). However, this does not lead to additional gains. Furthermore, this additional preprocessing step can only be carried out once the user prefix is known, additional processing time is required (in our implementation about 0.06 seconds on average).

6.6 Inside-Outside Pruning
Finally, by pruning low-scoring nodes in the parse forest, we can trade off faster processing time against matches with fewer edits. It is not so much a matter of if we make the choice, but what choice we make. Already in decoding, a pop limit determines how many nodes and edges we have in the parse forest, so the choice is made somewhere.

It is not entirely straightforward to gauge this trade-off. Consider the results in Table 4. With tighter thresholds (1, 0.5, 0.2, 0.1) for discarding edges, the number of zero-edit matches against the parse forest drops (741, 650, 499, 416), but processing time increases quite dramatically (0.54s, 0.07s, 0.01s, 0.01s). The improvements in processing time are especially dramatic when more edits are required — but note that more edits are required with such tighter beams.

In the previous experiments, we used a threshold of 0.5. The results suggest that this is a reasonable compromise.

7 Conclusions
We proposed two algorithms to solve the problem of matching partial translations against a parse for-
est, which arises in the context of interactive machine translation for syntax-based models. Of the two algorithms, the top-down search outperforms the bottom-up search.

We proposed three refinements. Our experiments show risk-free gains for removal of spurious ambiguity, but not for a more aggressive variant that normalizes all words in the parse forest that do not match any word in the user prefix. The third refinement is effective in trading off processing speed against match accuracy.

We plan to further validate this work with syntax-based models that use linguistic constituency labels, carry out experiments for other language pairs, especially German–English, where such syntax-based models have shown superior translation quality, and integrate the method into the CASMACAT workbench.