# The EUTRANS-I Speech Translation System\*

J.C.Amengual<sup>1</sup>, J.M.Benedí<sup>2</sup>, F.Casacuberta<sup>2</sup>, A.Castaño<sup>1</sup>, A.Castellanos<sup>1</sup>,

V.M.Jiménez<sup>1</sup>, D.Llorens<sup>1</sup>, A.Marzal<sup>1</sup>, M.Pastor<sup>2</sup>, F.Prat<sup>1</sup>, E.Vidal<sup>2</sup> and J.M.Vilar<sup>1</sup>

(1) Unidad Predepartamental de Informática, Campus Riu Sec, Universitat Jaume I, 12071 Castellón de la Plana (Spain)

(2) Depto. de Sistemas Informáticos y Computación and Instituto Tecnológico de Informática,

Universidad Politécnica de Valencia, 46022 Valencia (Spain)

**Abstract.** The EUTRANS project aims at using Example-Based approaches for the automatic development of Machine Translation systems —accepting text and speech input— for limited domain applications. During the first phase of the project, a speech translation system that is based on the use of automatically learnt Subsequential Transducers has been built. This paper contains a detailed and to a long extent self-contained overview of the transducer learning algorithms and system architecture, along with a new approach for using categories representing words or short phrases in both input and output languages. Experimental results using this approach are reported for a task involving the recognition and translation of sentences in the hotel reception communication domain, with a vocabulary of 683 words in Spanish. A translation word error rate of 1.97% is achieved in real time factor 2.7 in a Personal Computer.

Keywords: Speech translation, subsequential transducers, transducer learning, finite state models, grammatical inference

### 1. Introduction

Most of the current efforts to cope with the speech translation problem are based on the use of previously developed *text-input* translation systems relying on *knowledge-based* technology, which are *serially coupled* to the output of state-of-the-art word recognizer front-ends (Block, 1997; Bub *et al.*, 1997; Lavie *et al.*, 1997; Rayner and Carter, 1997).

In contrast, the EUTRANS project aims at building translation systems for text and speech input in limited domain applications by (i) using *example-based* techniques, and (ii) a tight integration of translation, syntactic and acoustic constraints into global models. In last years, example-based techniques have been showing their usefulness in translation systems; for instance, through a balanced combination with knowledge-based techniques (Nirenburg, 1995).

During the first phase of the project, a basic demonstration speech translation system has been developed that relies on a kind of finite state models known as Subsequential Transducers. Among the interesting properties of these models, we can remark:

- They can be automatically learnt from a text, sentence-aligned, bilingual corpus by efficient algorithms (Oncina, 1991; Oncina *et al.*, 1993; Oncina and Varó, 1996).
- They can be easily and efficiently used in conventional Continuous Speech Recognition systems so that, for each input acoustic sequence, the search for the optimal translation (and the corresponding input-language sentence) is guided by a model integrating

<sup>\*</sup> This work has been partially funded by the European Union and the Spanish CICYT, under grants IT-LTR-OS-20268 and TIC97-0745-C02, respectively. The EUTRANS project is being developed in two phases. This paper describes the approach adopted during the already finished first phase, that will be referred to as EUTRANS-I. The second phase is currently under development. See the project home page at http://hermes.zeres.de/Eutrans/



© 1999 Kluwer Academic Publishers. Printed in the Netherlands.

(i) the syntactic constraints of the input language, (ii) the information needed for the translation into the output language, and (iii) the syntactic constraints of the output language (Jiménez *et al.*, 1994; Jiménez *et al.*, 1995; Amengual et al., 1997a).

An important drawback of this approach is the large amount of bilingual examples required to learn useful translation models. In order to reduce the severity of this requirement, we show how appropriate models can be learnt from a categorized bilingual corpus in which words or short phrases (for instance, numbers, dates, or proper names) are replaced by adequate labels, thus simplifying the tasks that the learning algorithms have to tackle (Vilar *et al.*, 1995; Amengual et al., 1997a; Amengual *et al.*, 1997b).

The rest of the paper is organized as follows. In Section 2, we describe Subsequential Transducers and the learning algorithms. Although these algorithms have previously appeared in the literature (Oncina *et al.*, 1993; Oncina and Varó, 1996), for the sake of completeness we give them here in a more unified and hopefully understandable presentation. Section 3 is devoted to explain how to use categorization to reduce training data requirements. The integrated architecture of the EuTRANS-I speech translation system is presented in Section 4. Experimental results are reported in Section 5 and final conclusions are drawn in Section 6.

#### 2. The translation model and its basic learning algorithms

#### 2.1. NOTATION

Given an alphabet X,  $X^*$  is the free monoid of strings over X. First letters (a, b, c, ...) represent individual symbols of the alphabets and last letters (z, y, x, ...) represent strings of the free monoids. We refer to the individual elements of the strings by means of subindices, as in  $x = a_1 ... a_n$ . For any string  $x \in X^*$ , |x| denotes the *length* of x, and  $\lambda$  is the symbol for the string of length zero (empty string). Given two strings  $x, y \in X^*$ , xy denotes the *concatenation* of x and y.

If v is a string in  $X^*$  and  $L \subseteq X^*$ , then Lv(vL) denotes (in this paper) the set of strings  $xy \in L$  such that y = v(x = v). Hence,  $X^*v(vX^*)$  denotes the set of all strings of  $X^*$  that end (begin) with v, while  $\emptyset v = v\emptyset = \emptyset$  (the empty set). For  $u, v, w \in X^*$ , the suffix of v with regard to u is defined as  $u^{-1}v = w \Leftrightarrow v = uw$ , and the prefix of u with regard to v as  $uv^{-1} = w \Leftrightarrow u = wv$ . Given a set  $L \subseteq X^*$ , the longest common prefix of all the strings of L is defined as  $lcp(L) = v \Leftrightarrow L = vL$  and  $\forall u \in X^*, L = uL \Rightarrow |u| \leq |v|$ .

#### 2.2. FINITE STATE TRANSDUCERS

A Finite State Transducer (FST) is a finite state machine that accepts sentences from a given input language and produces associated sentences of an output language. It is composed of states and edges connecting them. Each edge has associated an input symbol and an output string. The parsing of an input sentence begins from a distinguished state (the initial state) and proceeds by consuming input symbols one by one. Every time an input symbol is matched following an adequate edge, the string associated to that edge is output and a new state is reached. This process continues on until the whole input is processed; then, additional output may be produced from the last state reached in the analysis of the input. An interesting class of FSTs are the Subsequential Transducers,



Figure 1. A simplified SST. The initial state has an arrow pointing to it and final states are marked by double-circling.

because there are well-known and efficient algorithms for inferring them from examples, as we will see below.

Formally, a FST is a tuple  $\tau = (X, Y, Q, q_0, E, \sigma)$  where X and Y are the *input* and output alphabets, Q is a finite set of states,  $q_0 \in Q$  is the *initial state*,  $E \subseteq Q \times X \times Y^* \times Q$  is a set of *edges*, and  $\sigma : Q \to Y^*$  is a state emission function<sup>1</sup>. Those states for which  $\sigma$  is defined are usually called *final states*. A Subsequential Transducer (SST) is a FST verifying that, if (p, a, y, q) and (p, a, y', q') belong to E, then y = y' and q = q' (the determinism condition). An example of a SST is shown in Figure 1.

Given a string  $x = a_1 \ldots a_n \in X^*$ , a sequence  $(p_0, a_1, y_1, p_1), \ldots, (p_{n-1}, a_n, y_n, p_n)$  is a path from  $p_0$  to  $p_n$  in  $\tau$  if  $(p_{i-1}, a_i, y_i, p_i) \in E$ ,  $i = 1, \ldots, n$ . When intermediate states are not important, a path will be expressed as  $(p_0, a_1 \ldots a_n, y_1 \ldots y_n, p_n)$ ,  $i = 1, \ldots, n$ . The set of all paths between two states  $p, q \in Q$  is denoted as  $\prod_{\tau} (p, q)$ . For every string  $x \in X^*$  such that  $\exists (q_0, x, y, q) \in \prod_{\tau} (q_0, q)$  and q is a final state we will say that  $(q_0, x, y, q)$  is a valid path, that x is accepted by  $\tau$  and that  $y\sigma(q)$  is a translation of x by  $\tau$ .

If  $\tau$  is a SST, the condition of determinism means that there can be no more than one valid path, and hence at most one translation, for a given input string. Therefore,  $\tau$  defines a function between an *input language*,  $L_I \subseteq X^*$ , and an *output language*,  $L_O \subseteq Y^*$ . Both  $L_I$  and  $L_O$  are regular languages and their corresponding automata are easily obtainable from the SST. In particular, an automaton for  $L_I$  can be obtained by eliminating the output of the edges and states, and considering the final state set of the automaton being the same as in the SST. A state is *useless* if it is not contained in any valid path. Useless states can be eliminated from a SST without changing the function it defines.

#### 2.3. INFERENCE OF SUBSEQUENTIAL TRANSDUCERS

In general, any subsequential transduction can be realized by several different SSTs. However, for each subsequential transduction, one of such transducers is the canonical SST for the transduction, which has the minimum number of states and is onward (Oncina, 1991; Oncina *et al.*, 1993). A SST  $\tau = (Q, X, Y, q_0, E, \sigma)$  is *onward* if  $\forall p \in Q - \{q_0\}$ ,

$$lcp(\{y_1 \dots y_n \sigma(q_n) \in Y^* | (p, x_1 \dots x_n, y_1 \dots y_n, q_n) \in E^*, \sigma(q_n) \neq \emptyset, n \ge 0\}) = \lambda.$$

In other words, the longest common prefix of the output strings in paths departing from p is  $\lambda$ . Equivalently, a SST is onward if, for each input string prefix, the output string associated to it by the transducer is the longest common prefix of the output strings (translations) corresponding to the input strings that begin with this input prefix.

In the following, basic algorithms which are formally guaranteed to infer the minimum onward SST which realizes a given subsequential transduction from a set of examples of the

<sup>&</sup>lt;sup>1</sup> In this paper, the term function refers to *partial* functions. We will use  $f(x) = \emptyset$  to denote that the function f is undefined for x.

transduction are described (Oncina, 1991; Oncina *et al.*, 1993; Oncina and Varó, 1996). In Section 2.3.1, functions that represent a set of training examples as simple forms of SSTs and produce compatible generalizations are presented. In Section 2.3.2, they are used to infer the minimum onward SST for a given total subsequential function. This algorithm generalizes the training examples by taking only translation structure into account. In Section 2.3.3, an algorithm which allows to incorporate syntactic constraints of the input and output languages into the translation network is described. The introduction of a specific input language model allows this last algorithm to infer the minimum onward SST for a given partial subsequential function. Finally, Section 2.3.4 explains how probabilistic information can be incorporated to the learnt SSTs. For interested readers, algorithms referenced in the following sections have been included in an appendix. Here, mainly intuitive and illustrative ideas of their behaviour are given.

#### 2.3.1. Generalization of a set of examples

Any unambiguous or single-valued finite set of samples (pairs of input-output strings)  $T \subset X^* \times Y^*$  can be immediately represented by means of a *Tree Subsequential Transducer* (TST). A TST,  $\tau = (Q, X, Y, q_0, E, \sigma)$ , for a given single-valued finite set of samples T, is a prefix tree acceptor for the input strings of T in which the output strings appear in the corresponding accepting states. Figure 2(a) shows an unambiguous set of examples, which have been drawn from room number translation from Spanish into English. The TST directly representing this sample set can be observed in Figure 2(b). Procedure Make\_TST (Algorithm 1, see appendix) can be used to build the TST of a given set T.

From this TST, an Onward Tree Subsequential Transducer (OTST), which also represents T, is built by producing the onward SST equivalent to the TST of T. Function  $Make\_OTST$  (Algorithm 2) presents a recursive procedure for obtaining the OTST for T from the TST for T. Mainly, this process consists in moving the longest common prefixes of the output strings, level by level, from the leaves of the tree toward the root. Figure 2(c) illustrates the result of this process. From the TST depicted in Figure 2(b), the longest output prefixes which are common among all paths departing from each state are moved towards the root of the tree. Uncommon output substrings remain at the highest level states and edges that they can reach in this recursive advancement process.

Note that the OTST obtained so far does not generalize the training set; that is, it is only able to translate strings that appear in the training set. A simple generalization of a set of samples T can be produced by merging two states of the OTST for T. The only property that these states must verify is that all paths departing from them which share the same sequence of input symbols must also share the same sequence of output strings. In this case, states are called *compatible* and they can be merged, resulting in a new SST which is a suitable generalization of the previous one.

The compatibility test requires sometimes pushing back some output string suffixes through the paths of one of the states (Algorithm 3). This operation is needed to help matching equal input symbols along with their output strings in the possibly common paths, and simply consist in moving a suffix of the output string of an edge to its following state and edges.

The function *Merge\_States* (Algorithm 4) merges two states of the SST and common paths departing from them. To this end, the compatibility of the output strings of the states is first tested. When this test succeeds, the edges of one of the states with input symbols not shared by the other state can be directly assigned to this last one. Also, edges

{  $(\lambda, \lambda)$ , (trescientos, three on oh), (seiscientos, six oh oh), (trescientos diez, three one oh), T(trescientos cincuenta, three five oh), (trescientos cincuenta y uno, three five one), (seiscientos cincuenta y siete, six five seven), (seiscientos ochenta, six eight oh), (seiscientos ochenta y cuatro, six eight four), (seiscientos veintitrés, six two three)

Sirag replacements  $ext{trescientos} / \lambda \\ ext{trescientos} / ext{three} \\ ext{selscientos} / ext{three} \\ \lambda \end{pmatrix}$ seiscient os / diezu/enta cincuenta /

cincuenta / five seven veintitres / X veintities / / eight ochenta / eight 1, three on on veintities / two three 2, six on on

3, three one on

4, three five oh 6, six eight on

7, six two three

11, three five one 12, six five seven

13, six eight four

fi

(a)





Figure 2. Simple SSTs representing a training set. (a) An unambiguous set of examples, T. (b) Tree Subsequential Transducer for T. (c) Onward Tree Subsequential Transducer for T.

sharing the same input symbols are adjusted on their output strings to recursively try to merge the destination states of the edges.

Such a recursive merging conveys merging of common paths starting at the initial pair of states. Recursion finishes successfully when all mergings are found compatible. Alternatively, it can be interrupted if the output strings of a pair of states are non compatible or if the output strings of a pair of equal input edges are non adjustable. This last case occurs if one of the edges has to be considered consolidated and the other one cannot be fitted to it. The notion of consolidated edge is related to the order in which pairs of states are merged and to the assumption that a merging of two states cannot modify the part of the transducer that has been being consolidated by previous compatible mergings.

Figure 3 illustrates the merging process for states 1 and 2 from the example OTST obtained in Figure 2(c). Output strings of these states are equal, thus the edge incoming state 2 is changed to reach state 1, edges outgoing state 2 are assigned to state 1 and state 2 is removed (Figure 3(a)). At this moment, two edges with input symbol "cincuenta" and different output strings depart from state 1. By pushing back the symbol "seven" in one edge, output strings of both edges are made equal allowing their merging along with their destination states (Figure 3(b)). States 4 and 5 can be merged because state 5 has no output string. Now, state 4 has two outgoing edges with input symbol "y" and different output strings. Pushing back symbols "one" and "seven" to the following edges makes the empty string to be the output string in both edges, which yields their merging and that of

}

#### PSfrag replacements uno / one siete / seven





Figure 3. Some steps of the merging process for states 1 and 2 from the Onward Tree Subsequential Transducer of Figure 2(c).

states 8 and 9 (Figure 3(c)). This merging process finishes successfully since all particular state and edge mergings have been found or have been made compatible.

#### 2.3.2. Inference of the translation structure

In order to guarantee the inference of a target subsequential transduction, the learning process requires the pairs of states of the initial OTST to be successively considered in a certain *order*. An appropriate order can be a *lexicographic order* of the input string prefixes. Notice that state numbering given to SSTs through Figures 2, 3, 4 and 5 follows such an order, which is obtained as a by-product of the TST construction (Algorithm 1), since states are named by means of the input prefixes that lead to them.

The Onward Subsequential Transducer Inference Algorithm (OSTIA) (Oncina, 1991; Oncina et al., 1993) is formally presented in Algorithm 5. It begins building the OTST of a finite single-valued training set  $T \subset X^* \times Y^*$  that receives as input. Then, OSTIA takes every state in lexicographic order, and tries to orderly merge each one with some other previous state. Merging of two states is made effective only if it is compatible. At the end, OSTIA returns an onward SST which is consistent with T; i.e., an onward SST which realizes T and a generalization derived from compatible mergings.

The class of *total* subsequential transductions can be identified in the limit from positive presentation of input-output pairs (Oncina, 1991; Oncina *et al.*, 1993). In other words, for any total subsequential transduction OSTIA will exactly obtain the minimum onward SST



Figure 4. Last steps in the execution of OSTIA after successfully merging states 1 and 2 (Figure 3(c)) from the Onward Tree Subsequential Transducer of Figure 2(c).

that realizes the subsequential transduction from a large enough set of input-output pairs of the function.

The behaviour of OSTIA on the example OTST of Figure 2(c) is outlined here below. It first tries to merge state 1 and state 0, which is not possible due to the distinct output strings of the states. Thus, state 1 remains as before, and now state 2 is considered. State 2 cannot be merged with state 0 either, but it can be merged with state 1 because they are compatible. Their detailed merging process was shown in Figure 3 and described in last section. Next, state 3 is considered to be merged on the SST obtained after merging states 1 and 2 (Figure 3(c)).

State 3 is compatible with state 0, provided that their output strings are equal and no path exists which can distinguish them. Therefore, state 3 is merged with state 0. Then, state 6 is found non compatible with states 0 and 1, due to their distinct output strings. However, state 6 can be merged with state 4, following a similar adjustment procedure to that previously described for states 1 and 2. The SST resulting from mergings of states 3 and 0 and states 6 and 4 is depicted in Figure 4(a).

Finally, states 7, 8, 11, 12 and 13 are all found compatible with state 0, yielding the SST presented in Figure 4(b) which is the onward SST consistent with T computed by OSTIA. Note that the obtained SST correctly associates output substrings to input symbols, which will allow it to appropriately translate other input strings not seen in the training. Such a behaviour generally appears in SSTs learnt by OSTIA, if a sufficiently large training set is available. However, although this behaviour is desirable in practice, it is not enough to adequately model practical translation tasks, as we discuss in next section.

2.3.3. Inference of translation models with given input and output syntactic constraints In practice, the SSTs learnt by OSTIA tend to very accurately translate *correct* input sentences, but also tend to accept and translate *incorrect* sentences producing meaningless results for them. This yields undesirable effects in case of noisy input, like the one obtained by optical character recognition, typing, or (of particular interest in our case) speech recognition. The SST of Figure 4(b) can accept and translate incorrect sentences, like "uno seiscientos ochenta y trescientos" which is translated as "one six eight three oh oh". This problem originates from the fact that, by state merging, OSTIA tends to overgeneralize the input and output languages as much as possible while accurately modeling the mapping from input to output sentences. That is, the finite state model of the input language underlying the learnt SST (the one resulting from removing the output strings associated to the edges) does not necessarily constitute a good input language model, and the same happens with the finite state model of the output language underlying the SST (the one resulting from removing the input symbols associate to the edges)

A possible way to overcome this over-generalization problem is to impose to the learning process the constraint that the learnt SSTs should not accept input sentences or produce output sentences which are not accepted by given models of the input (Domain) and output (Range) languages. If these constraints can be modeled by Deterministic Finite Automata (DFA), then learning can be carried out with a version of OSTIA called OSTIA-DR (Onward Subsequential Transducer Inference Algorithm with Domain and Range) (Oncina and Varó, 1996), which is given in Algorithm 6. It only differs from OSTIA in the test for deciding whether merging two states will be acceptable or not: OSTIA-DR will never merge SST states that correspond to different states in the DFA for the input language (Domain) or in the DFA for the output one (Range). Formally, let  $D = (Q_D, X, \delta_D, d_0, F_D)$  and  $R = (Q_R, Y, \delta_R, r_0, F_R)$  be two DFAs describing the Domain and Range of a subsequential function t, respectively. Given a SST  $\tau = (X, Y, Q, q_0, E, \sigma)$ , let  $(q_0, x_p, y_p, p)$  be a path in  $\Pi_{\tau}(q_0, p)$  and let  $(q_0, x_q, y_q, q)$  be a path in  $\Pi_{\tau}(q_0, q)$ . Then states p and q are only allowed to be merged if  $\delta_D(d_0, x_p) = \delta_D(d_0, x_q)$  and  $\delta_R(r_0, y_p) = \delta_R(r_0, y_q)$ . This test can be very efficiently implemented if the states and output symbols of the initial OTST are previously labelled with the corresponding states in the Domain and Range DFAs.

OSTIA-DR can make use of any kind of DFA models for Domain and Range. In particular, these models can be N-Testable Automata, which can be automatically learnt from examples (García and Vidal, 1990). N-Testable Automata are just the result of removing probabilistic information from stochastic N-Testable Automata which, in turn, constitute just a convenient structural way of representing the well known N-Gram models in terms of finite state machines (Vidal et al., 1995). Therefore, standard automata minimization algorithms can be applied to N-Testable Automata. Experience shows that using smaller, more compact Domain and Range models generally helps OSTIA-DR to produce better generalizations for a given amount of training data, and, hence, minimized models are generally used.

Figure 5 illustrates the result of executing OSTIA-DR with the unambiguous training set T of Figure 2(a). In order to provide a clear presentation, the figure only shows details related with the inclusion of a Domain model in the learning process. Figure 5(a) shows a minimized 2-Testable automaton for the input language of the transduction, which is used to label the states of the OTST for T, previously shown in Figure 2(c). The label added to each transducer state is simply the automaton state reached when the input prefix that leads to the state of the OTST is parsed through the Domain automaton. The resulting OTST with labelled states is presented in Figure 5(b). Inclusion of a Range model would imply labelling all symbols of the output strings of the OTST.

From this state labelled OTST, the generalization process carried out by OSTIA-DR yields the onward SST in Figure 5(c). Although its ordered merging process is similar to that of OSTIA, it can be observed that, in contrast with the transducer obtained by OSTIA (Figure 4(b)), states 3, 7, 8, 11, 12 and 13 have not been merged with state 0 since they have associated a different state label. Moreover, note that the structure of the onward SST learnt in this case is the same as that of the Domain automaton. In the



*Figure 5.* Key details of the learning process of OSTIA-DR. (a) Automaton for the Domain language. (b) Onward Tree Subsequential Transducer of Figure 2(c) labelled with automaton states. (c) Onward SST learnt by OSTIA-DR.

general case, translation constraints convey an extension of the structure determined by Domain and Range models.

### 2.3.4. Estimating transition probabilities

So far, only *structural* aspects involved in the learning of finite state translation models have been considered. However, in order to properly integrate these models with standard acoustic models to perform speech translation (see Section 4), not only structural, but also *probabilistic* aspects are important. Given that a SST is a deterministic model, optimal maximum likelihood estimates of the transition probabilities can be obtained by computing the relative frequency of use of each transition in the (deterministic) parsing of the text training sentences. This results in an Stochastic SST which models a joint probability distribution of input-output sentence pairs.

## 3. Using categories to reduce the amount of data required to learn the SSTs

An important drawback of the approach presented so far is that the required amount of training data rapidly grows with the complexity of the translation task to be modelled. Hence, some measures are required in order to apply this approach to non trivial tasks while keeping the number of needed examples affordable. Among a number of promising

approaches (Vidal, 1997), categorization has proved quite effective: we can try to simplify a given translation task by replacing some words or short phrases, both in the input and output languages, by adequate labels from a set of what we call *categories*. The basic idea consists in using the same category label to represent those words and expressions that play a similar role and, thus, are expected to appear in the same kind of contexts. Consider, for instance, the possibility of using a specific category for representing colors (black, pale blue, olive green ... ) and a different one for plane shapes (circle, square, ellipse, isosceles triangle ... ) in a translation task involving the description of visual scenes.

The approach for using categories together with SSTs presented in (Vilar *et al.*, 1995) proved to be useful in reducing the number of examples required for learning. However, this approach was not easily integrable in a speech recognition system and could not deal with categories including units larger than a word. For these reasons, in the EUTRANS-I project the approach was changed so that a single FST would comprise all the information for the translation, including elementary transducers for the categories. This can be achieved by following these steps:

- Definition of categories. Determine the set of categories.
- Corpus categorization. Replace words and short phrases in the corpus by their category labels.
- **Basic structure model learning**. Use the categorized corpus to train a model, which will be referred to as *initial SST*.
- **Category modelling**. For each category, learn a so-called category SST (cSST).
- **Category expansion**. Expand the edges in the initial SST corresponding to the different categories using their respective cSSTs. This expansion procedure, explained in more detail below, can introduce non-determinism, so the new model is a FST which will be referred to as *expanded FST*.

A general view of the process can be seen in Figure 6. The left part represents the elements involved in the learning of the expanded FST, exemplified with a single training pair. The right part of the diagram gives a schematic representation of the use of this transducer for the translation of speech input as will be explained in Section 4.

The category expansion step is a bit more complex than just substituting each categorylabelled edge by the corresponding cSST. It has to consider (i) how to insert the output of the cSST within the output of the initial transducer;(ii) how to deal with more than one final state in the cSST; and (iii) how to deal with cycles in the cSST involving its initial state.

Solving (i) is not trivial, since the translation of a category label can appear before or after the label has been seen in the input. For example, consider the transducer in Figure 7(a) and a Spanish sentence categorized as "me voy a \$HOUR", which corresponds to the categorized English one "I am leaving at \$HOUR". In our application, once "me voy a" is seen, the continuation can only be "\$HOUR", so the initial SST, before seeing this category label in the input, has already produced the whole output (including "\$HOUR"). Taking this into account, we decided to keep the output of the initial SST and to include there the information necessary for removing the category labels. To do this, the label for the category was considered as a variable that acts as a placeholder in the output sentence and whose contents are also fixed by an assignment appearing elsewhere within



Figure 6. General scheme of the treatment of categories in the learning and translation processes.

that sentence. In our example, the expected output for "me voy a las tres y media" could be "I am leaving at **\$HOUR \$HOUR=**[half past three]". This assumes that each category appears at most once within each sentence.

The expanded FST is then obtained by an iterative procedure which starts with the initial SST. For each edge whose input symbol is a category label, the following steps are performed:

- Eliminate the edge.
- Create a copy of the cSST corresponding to the category label.
- Add new edges linking the new cSST with the FST. These edges have to ensure that the output produced by the cSST is embraced between "c=[" and "]", c being the category label.
- Eliminate useless states.

More formally, given a FST  $\tau = (X, Y, Q, q_0, E, \sigma)$ , a cSST  $\tau_c = (X, Y, Q_c, q_{0c}, E_c, \sigma_c)$ , where we assume that  $\sigma_c(q_{0c}) = \emptyset$  (i.e., the initial state of the cSST is not a final one), and an edge  $(p, c, z, q) \in E$ , the edge expansion produces a new FST  $\tau' = (X, Y, Q \cup Q'_c, q_0, (E - (p, c, z, q)) \cup E'_c, \sigma')$  in which the new elements are: - The set  $Q'_c$ , disjoint with Q and such that there exists a bijection  $\phi: Q_c \to Q'_c$ .

- The new set of edges:

$$\begin{split} E'_{c} &= \left\{ (\phi(r), a, y, \phi(s)) \mid (r, a, y, s) \in E_{c} \right\} \\ &\cup \left\{ (p, a, zc = [y, \phi(s)) \mid (q_{0c}, a, y, s) \in E_{c} \right\} \\ &\cup \left\{ (\phi(r), a, y\sigma_{c}(s) ], q) \mid (r, a, y, s) \in E_{c} \land \sigma_{c}(s) \neq \emptyset \right\} \\ &\cup \left\{ (p, a, zc = [y\sigma_{c}(s) ], q) \mid (q_{0c}, a, y, s) \in E_{c} \land \sigma_{c}(s) \neq \emptyset \right\} \end{split}$$

- The new state emission function:

$$\sigma'(s) = \begin{cases} \sigma(s) & \text{if } s \in Q \\ \emptyset & \text{if } s \in Q'_c \end{cases}$$

Finally, the useless states that may appear during this construction are removed.

A simple example of the effects of this procedure can be seen in Figure 7. Drawing (a) depicts the initial SST, while (b) shows a cSST for the hours between one and three (in "o'clock" and "half past" forms) and the expanded FST is represented in (c).

Note that this procedure solves the problems derived from the cSST having multiple final states or cycles involving the initial state. The price to pay is the introduction of non-determinism in the model, which may lead to ambiguity. Transition probabilities can be straightforward estimated for unambiguous models as outlined in Section 2.3.4. Ambiguity, however, rises non trivial estimation problems which can be solved using different estimation techniques discussed in (Casacuberta, 1995; Casacuberta, 1996). An alternative approach, used in the experiments reported in Section 5.3, consists in independently estimating the transition probabilities of both the initial SST and all the cSSTs as outlined in Section 2.3.4, and then adequately combine these transition probabilities during the category expansion process.

### 4. The EuTrans-I integrated architecture for speech translation

If both a Continuous Speech Recognition (CSR) system and a text input translation device (for instance, a SST learnt by OSTIA) are available, we can build a speech translation system in a *decoupled* manner by simply feeding the text translator with the output of the CSR system (with, possibly, error correcting parsing in order to cope with noisy output from the recognizer). However, such a decoupled scheme has the disadvantage of not taking the syntactic restrictions underlying the transducer itself into account during the recognition process. Also, it does not seem to be an ideal solution when we have imperfect recognition and translation devices: the translation module would add to its own errors those produced by incorrectly recognized sentences which can not be correctly translated. Therefore the performance of the system resulting from serially coupling a recognition and a translation module should be expected to be lower than the performance of each one of them.

For this reason, a different, integrated architecture has been adopted in the EUTRANS-I system (Jiménez *et al.*, 1994; Jiménez *et al.*, 1995; Amengual et al., 1997a). The next sections describe the modeling levels and decoding algorithms of this system.



(c) Expanded FST.

Figure 7. An example of the categories expansion procedure.

### 4.1. Acoustic models

The most successful current approach to model the variability in speech production at the word or sub-word level uses (first order) Hidden Markov Models (HMMs) (Baker, 1975; Jelinek, 1976), which are composed by two stochastic processes: (i) a *Hidden process*, given by an homogeneous Markov chain, with discrete time parameter and finite set of states; and (ii) an observation process, given by output distributions associated with the states of the hidden process. The Markov chain is constituted by a finite set of states Q and a transition probability matrix defining, for each  $q, q' \in Q$ , the probability of visiting state q' immediately after state q. In an homogeneous Markov chain this probability is independent of time, and can be denoted as P(q'|q, h), where h is the HMM under consideration. In general, an additional probability distribution over the initial states is required. However, in CSR is usual to fix a state  $q_S$  with probability 1 of being initial. Denoting by O the space

of acoustic observations, the observation process is defined by a probability distribution (or a probability density function) associated with each state, P(o|q, h) being the probability of observing  $o \in O$  while in state q of h. A particular state  $q_F$  is the final state (which is non emitting and there are no transitions departing from it). The HMM can be regarded as a finite state machine that randomly generates sequences of observations as follows. Initially it departs from state  $q_1 = q_S$ . At time t it randomly outputs an observation  $o_t$ , according to the emission probability distribution, and moves from state  $q_t$  to state  $q_{t+1}$ , chosen randomly according to the transition probability distribution. The process stops when the final state  $q_F$  is reached. An external observer may have access to the generated sequence of observations  $o = o_1 o_2 \dots o_{|o|}$ , but the sequence of states is hidden to the observer.

The probability of a sequence of observations  $o \in O^*$  being produced by a HMM h is obtained by summing up, over all possible sequences  $q_1 \ldots q_{|o|+1}$  of |o| + 1 states with  $q_1 = q_S$  and  $q_{|o|+1} = q_F$ , the probability of visiting the sequence of states  $q_1 \ldots q_{|o|+1}$  times the probability of generating the sequence of observations o along the sequence of states  $q_1 \ldots q_{|o|+1}$ :

$$P(o|h) = \sum_{q_1 \dots q_{|o|+1}} \prod_{t=1}^{|o|} P(o_t|q_t, h) \cdot P(q_{t+1}|q_t, h)$$
(1)

In CSR systems, the acoustic observations are obtained after a preprocess of parameterization in which the relevant information is extracted from the speech signal acquired by the microphone. A HMM can be used for modeling the variability in the sequences of observations corresponding to different pronunciations of the same sub-word unit. The topology of the HMM (the structure of the graph of states and transitions with positive probability) can allow for modeling the different articulatory effects in the initial, central, and final parts of the sub-word unit. Also, time elongation or contractions can be modeled (for instance, by loops over the same state or transitions that skip over some state).

The emission probability distribution is usually a continuous parametric one, whose parameters are estimated from a large enough corpora of utterances. Possibly the most used distributions are mixtures of Gaussians:

$$P(o_t|q_t) = \sum_{c=1}^{N_t} w_{t,c} \cdot N(o_t|\mu_{t,c}, \Sigma_{t,c})$$
(2)

whose parameters are the number  $N_i$  of Gaussians per mixture, the weights  $w_{i,c}$ , the mean vectors  $\mu_{i,c}$ , and the covariance matrices  $\Sigma_{i,c}$  (the dependence with h has been omitted for simplicity). In order to reduce the amount of data required for a correct estimation of the parameters, some of them are usually fixed by hand or shared by different Gaussians, states or mixtures. Typical simplifications are fixing  $N_i$  to 1, assuming  $\Sigma_i$  diagonal, or assuming the same covariance matrix for all the Gaussians in the same mixture.

### 4.2. LEXICAL MODELS

In small-vocabulary tasks (for instance, recognition of sequences of digits), HMMs can be used to model vocabulary words. When the vocabulary size increases, the training data is usually not enough for individual modeling of each different word. In this case the usual

$$-\frac{1}{2}\left\{\frac{1}{2}\left\{\frac{1}{2}\left\{\frac{1}{2}\right\}-\frac{1}{2}\left\{\frac{1}{2}\left\{\frac{1}{2}\right\}-\frac{1}{2}\left\{\frac{1}{2}\left\{\frac{1}{2}\right\}-\frac{1}{2}\left\{\frac{1}{2}\left\{\frac{1}{2}\right\}-\frac{1}{2}\left\{\frac{1}{2}\left\{\frac{1}{2}\right\}-\frac{1}{2}\left\{\frac{1}{2}\left\{\frac{1}{2}\right\}-\frac{1}{2}\left\{\frac{1}{2}\left\{\frac{1}{2}\right\}-\frac{1}{2}\left\{\frac{1}{2}\left(\frac{1}{2}\right)\right\}-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)\right\}-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}{2}\right)-\frac{1}{2}\left(\frac{1}{2}\left(\frac{1}$$

*Figure 8.* HMM resulting after the integration in the SST of Figure 1 of the lexical and the acoustic models, for a case in which lexical models are simple concatenation of phonemes and acoustic models are 3-state context-independent HMMs.

approach is to use HMMs for modeling a small set of sub-word units. Typical sub-word units are phonemes. In this case, lexical models constitute the intermediate level between the acoustic and the language models, defining the mapping from words into sequences of phonemes. This also allows for easily adding new words to the vocabulary accepted by the system, without retraining the HMMs. The system can be adapted to different tasks by simply choosing different lexical and syntactic models, and optionally improving the acoustic models with training sentences of the task.

In a simple approach, each vocabulary word is represented by a single sequence of phonemes. In a more robust approach, speaker or dialectal variations are modeled so that the same word can be associated with different sequences of phonemes. This can be easily done by simply increasing the vocabulary with the different variants for each word. However, these alternatives can be more compactly represented by a stochastic finite state automaton, with phonemes associated with the edges in such a way that different paths correspond to different word pronunciations.

Lexical models are usually built by hand, or automatically by programs that implement phonological rules for the input language. Once the HMMs for phonemes have been trained, they can be joined or integrated together according to the lexical models, replacing each edge of the lexical models by the corresponding phoneme HMM. In this way a (bigger) HMM for each word in the vocabulary V is obtained, which would model the different sequences of acoustic observations in which the pronunciation of the word could result (Jelinek, 1976).

### 4.3. Syntactic and translation models

Sections 2 and 3 have described how to learn, from a (possibly categorized) bilingual corpus, a stochastic FST that represents, in an integrated way, (i) the syntactic constraints of the input language, (ii) the information needed for the translation into the output language, and (iii) the syntactic constraints of the output language (which help producing only well formed translations).

The HMMs for vocabulary words, possibly obtained from HMMs for sub-word units (see Section 4.2), can in turn be integrated within this stochastic FST, resulting in a large HMM which combines all the knowledge sources participating in the assignment of probabilities to sequences of acoustic observations. This is done by just substituting the edges of the FST by the HMMs for the corresponding input language words, as explained in (Jelinek, 1976) for the integration within language models. This is illustrated in Figure 8. In the resulting integrated HMM there are two types of states: non-emitting, corresponding to the states of the FST, and emitting, corresponding to the states of the original HMMs.

## 4.4. VITERBI DECODING

The speech translation problem can then be seen as a decoding problem that consists in finding the output sequence of words whose probability is maximum given the integrated HMM and given the sequence of acoustic observations. A common suboptimal approximation to solve the decoding problem is to find the path (sequence of states) in the integrated HMM whose probability is maximum, and then take the sequence of words associated to the edges traversed by that path. In our case, by following the edges in the optimal path, we can recover not only the (approximately) optimal sequence of words in the input language but also its corresponding translation by the FST.

The Viterbi algorithm (Viterbi, 1967; Forney, 1973) solves this problem of, given a HMM h and a sequence of observations o, find the state sequence q for which the a posteriori probability P(q|o, h) is maximum. It is a very efficient algorithm that just requires  $O(|h| \cdot |o|)$  time (where |h| is the number of edges in h and |o| is the length of o) and O(|h|) space. But even this can become computationally expensive when |h| is high. In this case a pruning technique known as *beam search* (Lowerre, 1976) can be performed: after processing each new observation  $o_t$ , those states whose cumulative score exceed the best current score by more than a given threshold, are pruned.

In order to reduce memory requirements, the integrated HMM does not need to be fully expanded in memory. For each new acoustic observation, only the successors of those states which are not pruned by the beam search need to be expanded (Ney *et al.*, 1987). A list linking the active states at time t can determine the possible active states at time t + 1. In the EuTRANS-I system, this is implemented by using two different beam search thresholds: one at the states of the FST (or inter-word transitions) and another at the acoustic states (or intra-word transitions). Choosing appropriate values for these thresholds can reduce both the temporal and spatial costs without significantly affecting the system performance. A structure of back-pointers is built linking the states of the FST which survive the beam search pruning. The optimal sentence hypothesis together with its translation is recovered at the end of the process from the inter-word transitions which constitute this structure.

In order to achieve close to real-time computation, each new acoustic observation obtained after the acoustic preprocessing is immediately supplied to the decoding module. This works in a so-called frame-synchronous (or left-to-right) manner, and so it can perform computation without waiting for the utterance to terminate.

### 5. Experiments

#### 5.1. The Traveler Task corpus

The *Traveler Task* corpus is a set of paired bilingual sentences (Spanish and English) that was built within the EUTRANS-I project. It is much more realistic that the one used in (Castellanos *et al.*, 1994), but, unlike other bilingual corpora such as the Hansards (Brown *et al.*, 1990), it is restricted to a limited domain.

The general framework established for the *Traveler Task* aims at covering usual sentences that can be needed by a traveler visiting a foreign country whose language he/she does not speak. This framework includes a great variety of different translation scenarios, and thus results appropriate for progressive experimentation with increasing complexity. In a first phase, the scenario has been limited to some human-to-human communication

Spanish:	¿Cuánto cuesta por día una habitación doble con pensión completa?
English:	How much does a double room with full board cost per day?
Spanish:	Quisiéramos reservar dos habitaciones para un día a nombre de Federico Mestre, por favor.
English:	We want to book two rooms for a day for Federico Mestre, please.
Spanish:	Por favor, dénos las llaves de la doscientos veintidós.
English:	Please give us the keys to room number two two two.
Spanish:	Por favor, ¿quieren pedirnos un taxi para la habitación trescientos diez?
English:	Will you ask for a taxi for room number three one oh for us, please?

Table I. Some examples of sentence pairs from the Spanish to English Traveler Task.

Table II. Main features of the Spanish to English text corpora.

	Spanish	English
Vocabulary size	683	514
Average sentence length	9.5	9.8
Test set perplexity	13.8	7.0

situations in the reception of a hotel: asking for rooms, wake-up calls, keys, the bill, a taxi and moving the luggage; asking information about rooms (availability, features, price); having a look at rooms, complaining about and changing them; notifying a previous reservation; signing the registration form; asking and complaining about the bill; notifying the departure; and other common expressions.

A small seed corpus was created from several guide books with sentences of common use for tourists. This corpus was used to help the design of the *Traveler Task* corpus, which was automatically built by using a set of Stochastic Syntax-Directed Translation Schemata (Gonzalez and Thomason, 1978) with the help of a data generation tool specially developed for the EUTRANS-I project. This software allows the use of several syntactic extensions to these schemata in order to express optional rules, permutation of phrases, concordance (of gender, number and case), etc. The use of automatic corpora generation was convenient due to time constraints of the first phase of the EUTRANS-I project, and cost-effectiveness. Moreover, this procedure allows to control the level of complexity of the task.

Some example pairs of the Spanish to English *Traveler Task* corpus are shown in Table I. Some features of this corpus can be seen in Table II. The test set perplexity has been computed by training a trigram model (with simple flat smoothing) using a set of 20,000 random sentences and computing the probabilities yielded by this model for a set of 10,000 independent random sentences. The lower perplexity of the output language derives from a design decision: multiple variants of the input sentences were introduced to account for different ways of expressing the same idea, but they were given the same translation. Finally, a multi-speaker speech corpus for the task was acquired. A total of 436 Spanish sentences were selected from the text corpus. They were divided into eleven sets: one common set consisting of 16 sentences, and ten sets of 42 sentences. Each one of twenty speakers (ten male and ten female) participating in the acquisition of this corpus, pronounced the common set and two out of the other ten, totalling 2,000 utterances, 15,360 words and about 90,000 phones. The sampling frequency was 16 kHz.

From this speech corpus, two sub-corpora were extracted:

- Training and adaptation (Trav TR): 16 speakers (eight male and eight female), 268 sentences, 1,264 utterances (approx. 11,000 words or 56,000 phones).
- Speaker independent test (*TravSI*): 4 speakers (two male and two female, not involved in *TravTR*), 84 sentences (not in *TravTR*), 336 utterances (approx. 3,000 words or 15,000 phones).

#### 5.2. TRANSLATION MODEL TRAINING EXPERIMENTS

First, we tested on the text corpus the capacity of OSTIA-DR for learning good translation models. This corpus was divided in a training set and a test one, with 490,000 and 10,000 pairs, respectively. Two sequences of FSTs were trained with increasing subsets of the training set:

- Without categories. For each subset of the training set, minimized 3-Testable Automata of the input and the output language were inferred and, using them as Domain and Range models, a SST was learnt by OSTIA-DR from the same subset.
- With categories. We chose categories which are easy to identify and that follow simple translation rules, so that the amount of special linguistic knowledge introduced is very low. Seven categories were used: masculine names, feminine names, surnames, dates, hours, room numbers, and general numbers. Simple scripts substituted the words in the categories by adequate labels. For example, the pair ("me voy el once de julio I am leaving on july the eleventh") would become ("me voy el \$DATE I am leaving on \$DATE"), where "\$DATE" would be the category label for dates. For each subset of the training set, and using again minimized N-Testable Automata as Domain and Range models, a FST was obtained following the approach described in Section 3 (but cSSTs were learnt from specific manually built corpora, instead of extracting them from the training subset by the categorizer).

Each model was tested using only those sentences in the test set that were not seen in training. This has been done because a model trained with OSTIA-DR is guaranteed to reproduce exactly those translations it has seen during learning. The performance was evaluated in terms of translation *Word Error Rate* (WER), which is the percentage of output words that have to be inserted, deleted and substituted in order to exactly match the corresponding expected translations.

The results can be seen in Table III. The columns labelled as "Different" and "Categ.", refer to the number of different sentences in the training set and the number of different sentences after categorization. As expected, the use of lexical categories had a major impact on the learning algorithm. The large increase in performance is a natural consequence of the fact that the categories help in reducing the total variability that can be found in the

Training pairs			Without catogories		With catogories			
Generated	Different	Categ.	WER	States	Edges	WER	States	Edges
10,000	6,791	5,964	60.72	$^{3,210}$	10,427	30.51	4,500	32,599
20,000	12,218	9,981	54.86	4,119	15,243	22.46	4,700	35,585
40,000	$21,\!664$	16,207	47.92	$^{5,254}$	22,001	13.70	4,551	34,879
80,000	38,438	25,665	38.39	6, 494	31,017	7.74	4,256	$37,\!673$
160,000	67,492	39,747	26.00	6,516	36,293	3.71	4,053	34,045
320,000	119,048	60,401	17.38	$6,\!249$	41,675	1.42	4,009	$33,\!643$
490,000	$168,\!629$	77,499	13.33	$5,\!993$	47,151	0.74	3,854	29,394

Table III. Text input results. Translation word error rates (WER) and sizes of the transducers for different number of training pairs.

corpora (although sentences do exhibit a great deal of variability, the underlying syntactic structure is actually much less diverse). They also have the advantage of allowing an easier extension in the vocabulary of the task with a lower negative effect on the performance of the models so obtained.

#### 5.3. Speech input experiments

Spanish to English speaker independent speech translation experiments were performed using the integrated architecture described in Section 4, with the following models:

- Acoustic level. Each one of 25 context-independent Spanish phonemes (including two types of silence: initial and final) was modeled by a continuous-density HMM with three emitting states and a left-to-right topology with loops in the emitting states. The emission distribution of each state was a mixture of Gaussians. The HTK Hidden Markov Model Toolkit V1.5 (Young *et al.*, 1993) was used to estimate the parameters of these HMMs from the union of two corpora: the 1,264 utterances in the *TravTR* sub-corpus, and an additional set of 1,530 utterances (by 9 speakers, 4 male and 5 female) from a different, quasi-phonetically-balanced corpus. This speech material was processed to obtain, each 10 msecs, 10 cepstral coefficients of a Mel-filter bank plus the energy and the corresponding first and second derivatives. The final models had a total of 2,462 Gaussians.
- Lexical level. Each word was represented by a simple chain of phones, which was automatically derived using standard rules from the Spanish Phonetics.
- Syntactic and translation level. The best of the transducers with categories obtained in the Spanish to English text experiments was used after estimating the transition probabilities as commented in Section 3.

After these models were trained, the system was used to recognize and translate into English the 336 Spanish utterances of the *TravSI* sub-corpus.

A series of experiments were then carried out in order to tune the beam search thresholds. The results in Table IV show how they can be adjusted to find an adequate tradeoff between accuracy and computing time. For instance, a Translation WER of 1.97 % can be achieved with a real time factor of just 2.7. When translation accuracy is the main

Language Model Beam Width	Acoustic Model Beam With	Recognition Word Error Rates	Translation Word Error Rates	Real Time Factor
	50	37.94%	40.37%	0.7
100	100	5.19%	5.13%	1.4
100	200	2.15%	2.14%	2.6
	400	2.15%	2.14%	4.9
	50	37.94%	40.37%	0.7
200	100	5.19%	5.13%	1.3
200	200	2.05%	1.97%	2.7
	400	1.98%	1.83%	5.8

Table IV. Speech input results. Effect of the beam widths in the recognition and translation time and accuracy.

Table V. Comparison between the integrated scheme (corresponding to the last row of Table IV) and the decoupled scheme (recognition using a trigram, the same acoustic and lexical models and the same beam search thresholds; and translation using a SST learnt without Domain and Range).

Approach	Recognition Word	Translation Word	Real Time
	Error Rates	Error Rates	Factor
${ m Decoupled}$ Integrated	$2.15~\% \\ 1.98~\%$	$3.54~\% \\ 1.83~\%$	$5.7 \\ 5.8$

concern, wider thresholds can be used in the search to achieve a Translation WER of 1.83%, but with a real time factor of 5.8. These results were obtained on a Intel Pentium 166Mhz Personal Computer running Linux, without resorting to any type of specialized hardware or signal processing device, and required no more than 16 Mb of memory.

The proposed integrated architecture was also compared against a decoupled scheme in which, instead of integrating the input (and output) language constraints in the learnt transducers, recognition was performed with the stochastic 3-Testable Automata (equivalent to a trigram) of the input language, and then the output of the recognizer was translated by a SST learnt by OSTIA (without Domain or Range constraints) from the same categorized corpus. The same acoustic and lexical models were used. The results in Table V confirm those reported by (Jiménez *et al.*, 1995) for a simpler translation task: the integrated approach not only offers better translation but also better recognition performance; that is, not only the input language constraints but also the translation and output language constraints for the application domain can help in finding which was the uttered sentence and also its corresponding translation. It is also worth noting the relation of recognition and translation WER in both approaches. In the decoupled approach, recognition errors are amplified by the translation process. In contrast, the integrated approach, taking advantage of the lower perplexity of the output language (see Table II), obtains a translation WER lower than the recognition WER.

21

Finally, we should remark that the results presented here are better that those reported in (Amengual *et al.*, 1997b), which were obtained on a HP-9735 workstation, reflecting improvements in our acoustic models: the set of phones considered, the topology of the HMMs and the HMM training software have been changed.

### 6. Conclusions

Finite State Transducers can be used as the basis of speech translation systems for limited domains. These models can be automatically learnt from examples, and the learning process can be improved by means of categories using the approach detailed in this paper. This approach has been tested in a task involving the recognition and translation of utterances in the hotel reception communication domain, with a vocabulary of 683 words in Spanish. Experiments with text input show that using categories significantly reduces the number of examples required for achieving good models. In experiments with speech input, a 1.97% translation word error rate is achieved in real time factor 2.7 in a Personal Computer without using specialized hardware. It is worth noting that there is a clear tradeoff between computing time and accuracy. For off-line operation, a different configuration can provide improved translation performance at the cost of increasing the real time factor (a 1.83% translation word error rate has been achieved in real time factor 5.8).

Automatically learning translation models from examples can lead to systems that can be easily modified and adapted to a great variety of tasks and language pairs, provided that the required corpora are available. Therefore this is an approach that clearly is worth continuing to explore. Our current work concentrates in further reducing the number of examples necessary for training the translation models, by reordering the words in the translations (Vilar *et al.*, 1996) or using new inference algorithms (Vilar, 1998). We are also exploring techniques for automatic bilingual categorization, and error correcting techniques for dealing with more spontaneous input. Finally, our system is in continuous development in order to deal with increasing vocabulary size and to get closer to other state-of-the-art CSR systems, so that our results could be more fairly compared to those of other spoken language translation projects.

#### Acknowledgements

The authors wish to thank the anonymous reviewers who helped to improve the quality and the presentation of this paper.

#### References

Amengual, J. C.; J. M. Benedí; K. Beulen; F. Casacuberta; A. Castaño; A. Castellanos; V. M. Jiménez; D. Llorens; A. Marzal; H. Ney; F. Prat; E. Vidal; and J. M. Vilar: 1997, "Speech Translation based on Automatically Trainable Finite-State Models", in G. Kokkinakis, N. Fakotakis, and E. Dermatas (eds.), *Proceedings of the EuroSpeech*, Rhodes, Greece, ESCA, pp. 1439–1442.

Amengual, J. C.; J. M. Benedí; F. Casacuberta; A. Castaño; A. Castellanos; D. Llorens; A. Marzal; F. Prat; E. Vidal; and J. M. Vilar: 1997, "Using Categories in the Eutrans System", in Steven Krauwer, Doug Arnold, Walter Kasper, Manny Rayner and Harold Somers (eds.), Proceedings of the Spoken Language Translation Workshop, Madrid, Spain, pp. 44-53.

- Baker, J. K.: 1975, "The Dragon System an Overview" IEEE Trans. on Acoustics, Speech, and Signal Processing ASSP-23(1), 24-29.
- Brown, Peter F.; John Cocke; Stephen A. Della Pietra; Vincent J. Della Pietra; Frederick Jelinek; John D. Lafferty; Robert L. Mercer; and Paul S. Roossin: 1990, "A Statistical Approach to Machine Translation" Computational Linguistics 16(2) 79–85.
- Block, Hans Ulrich:1997, "The Language Components in Vermobil", in Proceedings of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Munich, Germany, pp. 79-82.
- Bub, Thomas; Wolfgang Wahlster; and Alex Waibel: 1997, "Vermobil: The Combination of Deep and Shallow Processing for Spontaneous Speech translation", in *Proceedings of the IEEE Int. Conf. on* Acoustics, Speech, and Signal Processing, Munich, Germany, pp. 71-74.
- Casacuberta, F.: 1995, "Probabilistic Estimation of Stochastic Regular Syntax-Directed Translation Schemes" in A. Calvo and R. Medina (eds.), Pattern Recognition and Image Analysis, Preprints of the VI Spanish Symposium on Pattern Recognition and Image Analysis, PP. 201-207.
- Casacuberta, F.: 1996, "Maximum Mutual Information and Conditional Maximum Likelihood Estimations of Stochastic Syntax-Directed Translation Schemes", in L. Miclet and C. de la Higuera (eds.), Grammatical Inference: Learning Syntax from Sentences, LNAI, vol. 1147, Springer. pp. 282-291.
- Castellanos, A.; I. Galiano; and E. Vidal:1994, "Application of OSTIA to Machine Translation Tasks", in Rafael C. Carrasco and José Oncina (eds.), Grammatical Inference and Applications, Lecture Notes in Computer Science, vol. 862, Springer-Verlag, pp. 93-105.
- Forney, G. D.: 1973, "The Viterbi Algorithm", Proc. IEEE 61(3), 268-278.
- García, P. and E. Vidal:1990, "Inference of K-Testable Languages in the Strict Sense and Applications to Syntactic Pattern Recognition", IEEE Trans. Pattern Analysis Machine Intelligence, 12(9), 920–925.
- Gonzalez, Rafael C.; and Michael G. Thomason: 1978, Syntactic Pattern Recognition: An Introduction, Addison-Wesley, Reading, Massachusetts.
- Jelinek, F.: 1976, "Continuous Speech Recognition by Statistical Methods", Proc. IEEE 64(4), pp. 532-556.
- Jiménez, V. M.; A. Castellanos; and E. Vidal:1995, "Some Results with a Trainable Speech Translation and Understanding System", in *Proceedings of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Detroit, Michigan USA, pp. 113-116.
- Jiménez, V. M.; E. Vidal; J. Oncina; A. Castellanos; H. Rulot; and J. A. Sánchez: 1994, "Spoken-Language Machine Translation in Limited-Domain Tasks", in H. Niemann, R. de Mori, and G. Hanrieder (eds.), Progress and Prospects of Speech Research and Technology. Infix, pp. 262-265.
- Lavie, Alon; Alex Waibel; Lori Levin; Michael Finke; Donna Gates; Marsal Gavaldà; Torsten Zeppenfeld; and Puming Zhan:1997, "JANUS-III: Speech-to-Speech Translation in Multiple Languages", Proceedings IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Munich, Germany, pp. 99-102.
- Lowerre, B. T. : 1976, "The HARPY Speech Recognition System". PhD dissertation, Carnegie Mellon University. USA.
- Ney, H.; D. Mergel; A. Noll; and A. Paeseler: 1987, "A Data-Driven Organization of the Dynamic Programming Beam Search for Continuous Speech Recognition", in *Proceedings of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 833-836.
- Nirenburg, Sergei (editor): 1995, "The PANGLOSS Mark III Machine Translation System" (a joint technical report by NMSU CRL, USC ISI and CMU CMT), Technical Report CMU-CMT-95-145, Carnegie Mellon University, USA.
- Oncina, José: 1991, "Aprendizaje de Lenguajes Regulares y Funciones Subsecuenciales [Learning Regular Languages and Subsequential Functions]". PhD dissertation, Univ. Politécnica de Valencia, Spain.
- Oncina, José; Pedro García; and Enrique Vidal:1993, "Learning Subsequential Transducers for Pattern Recognition Interpretation Tasks", *IEEE Trans. Pattern Analysis Machine Intelligence* **15**(5), 448-458.
- Oncina, José and Miguel Ángel Varó: 1996, "Using Domain Information during the Learning of a Subsequential Transducer", in Laurent Miclet and Colin de la Higuera (eds.), Grammatical Inference: Learning Syntax from Sentences, Lecture Notes in Computer Science, vol. 1147, Springer, pp. 301-312,
- Rayner, Manny and David Carter:1997, "Hybrid Language Processing in the Spoken Language Translator" Proceedings IEEE Int. Conf. on Acoustics, Speech. Signal Processing, Munich, Germany, pp. 107-110.
- Vidal, Enrique; Francisco Casacuberta; and Pedro García: 1995: "Grammatical Inference and Automatic Speech Recognition", in A. J. Rubio and J. M. López (eds.), Speech Recognition and Coding: New Advances and Trends, vol. F147 of NATO ASI, Springer-Verlag, pp. 174–191.
- Vidal, E.:1997, "Finite-State Speech-to-Speech Translation" in *Proceedings of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Munich, Germany.

- Vilar, J. M.; V. M. Jiménez; J. C. Amengual; A. Castellanos; D. Llorens; and E. Vidal: 1996, "Text and Speech Translation by means of Subsequential Transducers", *Natural Language Engineering* 2(4), 351-354.
- Vilar, J. M.; A. Marzal; and E. Vidal: 1995, "Learning Language Translation in Limited Domains Using Finite-State Models: Some Extensions and Improvements", in *Proceedings of the EuroSpeech*, Madrid, Spain, ESCA, pp. 1231-1234.
- Vilar, J. M.: 1998, "Aprendizaje de Traductores Subsecuenciales para su Empleo en Tareas de Dominio Restringido [Learning Subsequential Transducers for Limited Domain Tasks]", PhD dissertation, Universidad Politécnica de Valencia, Spain.
- Viterbi, A. J.:1967, "Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm", *IEEE Trans. on Information Theory* **13**, 260–269.
- Young, S. J.; P. C. Woodland; and W. J. Byrne: 1993: "HTK: Hidden Markov Model Toolkit V1.5", Cambridge University Engineering Department and Entropic Research Laboratories Inc.

#### Appendix. Transducer Learning Algorithms

In this appendix, all algorithms described and referenced in Section 2.3 are formally presented. Both learning algorithms, Onward Subsequential Transducer Inference Algorithm (OSTIA) and Onward Subsequential Transducer Inference Algorithm with Domain and Range (OSTIA-DR), have been structured by means of common functions, which are first introduced.

A first function (Algorithm 1) is used to build a *Tree Subsequential Transducer* (TST), which is a prefix tree representation for the input strings of a given unambiguous set of training samples. In this prefix acceptor, each output string is associated to the accepting state of the corresponding input one.

Algorithm 1. Make\_TST

```
 \begin{array}{l} \text{Input: } T \subseteq X^* \times Y^* \; / \; \forall (x, y), (x', y') \in T, \; x = x' \Rightarrow y = y' \\ \text{Output: } \tau = (X, Y, Q, q_0, E, \sigma), \; \text{a TST for } T \\ Q := \{\lambda\}; \;\; q_0 := \lambda; \;\; E := \emptyset; \\ \text{for all } (x, y) = (a_1 \dots a_{|x|}, b_1 \dots b_{|y|}) \in T \; \textbf{do} \\ \; \forall i \in \{1, \dots, |x|\}, \;\; Q := Q \cup \{a_1 \dots a_i\}; \\ \; \forall i \in \{1, \dots, |x|\}, \;\; E := E \cup \{(a_1 \dots a_{i-1}, a_i, \lambda, a_1 \dots a_i)\}; \\ \; \forall i \in \{1, \dots, |x| - 1\}, \;\; \sigma(a_1 \dots a_i) := \emptyset; \\ \; \sigma(a_1 \dots a_{|x|}) := y; \\ \text{end for} \\ \text{return}(\tau); \end{array}
```

Then, a second function (Algorithm 2) obtains an Onward Tree Subsequential Transducer (OTST). Starting from the previous TST, the longest common prefixes of the output strings are recursively moved, level by level, from the leaves toward the root of the tree.

Next function (Algorithm 3) takes an edge of the transducer and an output suffix (of the output string of the edge), and moves this suffix from the edge to its following state and edges. This operation is used to try matching paths in the transducer that could be the same.

The last function (Algorithm 4) attempts to merge two states of the transducer and paths departing from them. To this end, it recursively tests the compatibility of paired states and edges. Recursion finishes successfully when all mergings are found compatible. Algorithm 2.  $Make\_OTST$ Input:  $\tau = (X, Y, Q, q_0, E, \sigma)$  a TST for a given T;  $x \in Q$ Output:  $\tau' = (X, Y, Q, q_0, E', \sigma')$ , an OTST for T  $\tau' := \tau$ ; for all  $(x, a, \lambda, xa) \in E'$  do  $\tau' := Make\_OTST(\tau', xa)$ ;  $z := lcp(\{y \in Y^* \mid (xa, b, y, xab) \in E'\} \cup \{\sigma'(xa)\})$ ;  $\forall (xa, b, y, xab) \in E', E' := (E' - \{(xa, b, y, xab)\}) \cup \{(xa, b, z^{-1}y, xab)\}$ ;  $\sigma'(xa) := z^{-1}\sigma'(xa)$ ;  $E' := (E' - \{(x, a, \lambda, xa)\}) \cup \{(x, a, z, xa)\}$ ; end for return $(\tau')$ ;

Algorithm 3. Push\_Back

 $\begin{array}{ll} \textbf{Input:} \ \tau = (X,Y,Q,q_{0},E,\sigma); & (r,a,y,r') \in E; & v \in X^{*} \ / \ y = uv, with \ u \in X^{*} \\ \textbf{Output:} \ \tau' = (X,Y,Q,q_{0},E',\sigma') \\ \tau' := \tau; \\ \forall (r',b,z,r'') \in E', \quad E' := (E' - \{(r',b,z,r'')\}) \cup \{(r',b,vz,r'')\} \\ \textbf{if} \ \sigma'(r') \neq \emptyset \ \textbf{then} \ \sigma'(r') := v\sigma'(r'); \\ E' := (E' - \{(r,a,y,r')\}) \cup \{(r,a,yv^{-1},r')\}; \\ \textbf{return}(\tau'); \end{array}$ 



```
Input: \tau = (X, Y, Q, q_0, E, \sigma); q, r, s \in Q
Output: compatibles \in {true, false}; \tau' = (X, Y, Q', q_0, E', \sigma')
\tau' := \tau; \quad compatibles := \mathbf{false};
if \sigma'(r) = \emptyset or \sigma'(s) = \emptyset or \sigma'(r) = \sigma'(s) then
     if \sigma'(r) = \emptyset then \sigma(r) := \sigma(s);
      for all (s, a, z, s') \in E' do
           if (r, a, y, r') \notin E' then
                  E' := (E' - \{(s, a, z, s')\}) \cup \{(r, a, z, s')\};
            else
                  if r' \prec q and y \notin Pr(z) then return(false, \tau'),
                  u := lcp(\{y, z\});
                  \tau' := Push_Back(\tau', (r, a, y, r'), u^{-1}y);
                  \tau' := Push_Back(\tau', (s, a, z, s'), u^{-1}z);
                  (compatibles, \tau') := Merge\_States(\tau', q, r', s');
                  if not compatibles then return(false, \tau')
            end if
      end for
      Q' := Q' - \{s\}; compatibles := true;
end if
return(compatibles, \tau');
```

To guarantee the inference of target subsequential transductions, learning algorithms try state merging following a *lexicographic order*,  $\prec$ , which is obtained from the TST construction, since states are named by the input prefixes leading to them. Given a SST  $\tau = (X, Y, Q, q_0, E, \sigma)$  such that  $Q \subseteq X^*$ , next functions implement such a state ordering:

first( $\tau$ ) returns  $r = \lambda \in Q / \forall r' \in Q, r \preceq r';$ last( $\tau$ ) returns  $r \in Q / \forall r' \in Q, r' \preceq r;$  and

 $next(\tau, s)$ , with  $s \in Q$ , returns  $r \in Q / \forall r' \in X^*$ ,  $s \prec r' \prec r \rightarrow r' \notin Q$ .

The Onward Subsequential Transducer Inference Algorithm (OSTIA) (Algorithm 5) infers SSTs using only the translation constraints reflected in the training set.

```
Algorithm 5. OSTIA
```

 $\begin{array}{ll} \textbf{Input:} \ T \subset X \times Y, \ \text{single-valued finite set of samples} \\ \textbf{Output:} \ \tau = (X,Y,Q,q_0,E,\sigma), \ \text{Onward SST consistent with } T \\ \tau := TST(T); \quad \tau := OTST(\tau,\lambda); \ q := first(\tau); \\ \textbf{while} \ q \prec last(\tau) \ \textbf{do} \\ q := next(\tau,q); \quad p := first(\tau); \quad compatibles := \textbf{false}; \\ \textbf{while not compatibles and } p \prec q \ \textbf{do} \\ \tau' := \tau; \\ \forall (r,a,w,q) \in E', \quad E' := (E' - \{(r,a,w,q)\}) \cup \{(r,a,w,p)\}; \\ (compatibles,\tau') := Merge\_States(\tau',q,p,q); \\ \textbf{if compatibles then } \tau := \tau'; \\ p := next(\tau,p); \\ \textbf{end while} \\ \textbf{end while} \end{array}$ 

The Onward Subsequential Transducer Inference Algorithm with Domain and Range (OSTIA-DR) (Algorithm 6) infers SSTs using both syntactic and translation constraints.

```
Algorithm 6. OSTIA-DR
```

```
Input: T \subset X \times Y, single-valued finite set of samples;
                D = (Q_D, X, \delta_D, d_0, F_D), a DFA representing the Domain language;
                R = (Q_R, Y, \delta_R, r_0, F_R), a DFA representing the Range language;
Output: \tau = (X, Y, Q, q_0, E, \sigma), Onward SST consistent with T, D and R
\tau := TST(T); \quad \tau := OTST(\tau); \quad q := first(\tau);
while q \prec last(\tau) do
       q := next(\tau, q); \quad p := first(\tau); \quad compatibles := false;
      while not compatibles and p \prec q do
            Let (q_0, x_p, y_p, p) \in \Pi_{\tau}(q_0, p) and (q_0, x_q, y_q, q) \in \Pi_{\tau}(q_0, q);
           if \delta_D(d_0, x_p) = \delta_D(d_0, x_q) and \delta_R(r_0, y_p) = \delta_R(r_0, y_q) then
                  \tau' := \tau;
                  \forall (r, a, w, q) \in E', \quad E' := (E' - \{(r, a, w, q)\}) \cup \{(r, a, w, p)\};
                  (compatibles, \tau') := Merge\_States(\tau', q, p, q);
                  if compatibles then \tau := \tau';
            end if
            p := next(\tau, p);
      end while
end while
```

mt98revision.tex; 3/09/1999; 17:54; p.26