Learning Finite-State Models for Language Understanding*

David Picó, Enrique Vidal

Institut Tecnològic d'Informàtica
Universitat Politècnica de València, 46020 València, SPAIN
e-mail: {dpico,evidal}@iti.upv.es

Abstract. Language Understanding in limited domains is here approached as a problem of language *translation* in which the target language is a *formal* language rather than a natural one. Finite-state transducers are used to model the translation process. Furthermore, these models are automatically learned from training data consisting of pairs of natural-language/formal-language sentences. The need for training data is dramatically reduced by performing a two-step learning process based on lexical/phrase categorization. Successful experiments are presented on a task consisting in the "understanding" of Spanish natural-language sentences describing dates and times, where the target formal language is the one used in the popular Unix command *"at"*.

1 Introduction

Language Understanding (LU) has been the focus of much research work in the last twenty years. Many classical approaches typically consider LU from a linguistically motivated, generalistic point of view. Nevertheless, it is interesting to note that, in contrast with some general-purpose formulations of LU, many applications of interest to industry and business have *limited domains*; that is, lexicons are of small size and the semantic universe is limited. If we restrict ourselves to these kinds of tasks, many aspects of system design can be dramatically simplified.

In fact, under the *limited-domain* framework, the ultimate goal of a system is to *drive the actions* associated to the meaning conveyed by the sentences issued by the users. Since actions are to be performed by machines, the understanding problem can then be simply formulated as *translating* the *natural language* sentences into *formal sentences* of an adequate (computer) command language in which the actions to be carried out can be specified. For example, "understanding" natural language (spoken) queries to a database can be seen as "translating" these queries into appropriate computer-language code to access the database. Clearly, under such an assumption, LU can be seen as a possibly simpler case of Language Translation in which the output language is *formal* rather than *natural*.

Hopefully, these simplifications can lead to new systems that are more compact and faster to build thant those developed under more traditional paradigms. This would entail i) to devise *simple* and easily understandable models for LU, ii) to formulate LU as some kind of *optimal search* through an adequate structure based on these models, and iii) to develop techniques to actually *learn* the LU models from training data of each considered task. All these requirements can be easily met through the use of *Finite-State Translation Models*.

The capabilities of Finite-State Models (FSM) have been the object of much debate in the past few years. On the one hand, in the Natural Language (NL) community, FSMs have often

* Work partially supported by the Spanish CICYT under grant TIC-0745-CO2

been ruled out for many NL processing applications, including LU, even in limited domains. Recently, many NL and Computational Linguistic researchers are (re-)considering the interesting features of FSMs for their use in NL processing applications [10].

Undoubtedly, the most attractive feature of FSMs consists in their simplicity: representation is just a matter of setting a network of nodes and links in memory, and parsing can be simply carried out by appropriately following the links of this network, according to the observed input data. More specifically, as it is well known, using Viterbi-like techniques, computing time for parsing is *linear* with the length of the data sequence to be parsed and, using adequate techniques, such as *beam search*, it can be easily made independent on the size of the network in practice. [2]

Simple as they are, FSMs generally need to be huge in order to be useful approximations to complex languages. For instance, an adequate 3–Gram Language Model for the language of the Wall Street Journal is a FSM that may have as many as 20 million edges [23]. Obviously, there is no point in trying to manually build such models on the base of a priori knowledge about the language to be modeled: the success lies in the possibility of automatically learning them from large enough sets of training data [8, 23]. This is also the case for the finite-state LU models used in the work presented in this paper [15, 24, 26].

2 Subsequential Transduction

The following definitions follow closely those given in Berstel [4], with some small variations for the sake of brevity. A *Finite State Transducer* (FST) is a six tuple $\tau = (Q, X, Y, q_0, Q_F, E)$, where Q is a finite set of states, X, Y are input and output alphabets, $q_0 \in Q$ is an initial state, $Q_F \subset Q$ is a set of final states and $E \subset Q \times X^* \times Y^* \times Q$ are the edges or transitions. The output associated by τ to an input string, x, is obtained by concatenating the output strings of the edges of τ that are used to parse the successive symbols of x.

One problem of using Finite State Transducers in our framework is that the problem of learning of general Finite State Transducers is at least as hard as the problem of learning a general Finite State Automaton, which is well known to be probably intractable. So we need a less general type of transducers. A *Sequential Transducer* (ST) is a five tuple $\tau = (Q, X, Y, q_0, E)$, where $E \subset Q \times X \times Y^* \times Q$ and all the states are accepting ($Q_F = Q$) and deterministic; i.e., $(q, a, u, r), (q, a, v, s) \in E \Rightarrow (u = v \wedge r = s)$. An important restriction of STs is that they preserve increasing length input-output prefixes; i.e., if t is a sequential transduction, then $t(\lambda) = \lambda, t(uv) \in t(u)Y^*$, where λ is the empty or *Nil* string.

While the use of sequential translation models has proved useful for LU in a number of rather simple tasks [21, 19, 20, 26], the limitations of this approach clearly show up as the conceptual complexity of the task increases. The main concern is that the required sequentiality assumption often prevents the use of "semantic languages" that are expressive enough to correctly cover the underlying semantic space and/or to actually introduce the required semantic constraints. As we will see below, input-output sequentiality requirements can be significantly relaxed through the use of Subsequential Transduction. This would allow us to use more powerful semantic languages that need only be *subsequential* with the input.

A *Subsequential Transducer* (SST) is defined to be a six-tuple $\tau = (Q, X, Y, q_0, E, \sigma)$, where $\tau' = (Q, X, Y, q_0, E)$ is a Sequential Transducer and $\sigma : Q \rightarrow Y^*$ is a partial *state output function* [4]. An output string of τ is obtained by concatenating $\sigma(q)$ to the usual sequential output string, $\tau'(x)$, where q is the last state reached with the input x. Examples of SSTs are shown in Fig.1.

Two SSTs are equivalent if they perform the same input-output mapping. Among equivalent SSTs there always exists one that is *canonical*. This transducer always adopts an *"onward"* form, in which the output substrings are assigned to the edges in such a way that they are as "close" to the initial state as they can be (see Oncina *et al.*, 1993 [15], Reutenauer, 1990 [22]; for a recent reelaboration of these concepts see Mohri, 1997 [13]). On the other hand, any finite (training) set of input-output pairs of strings can be properly represented as a *Tree Subsequential Transducer* (TST), which can then be easily converted into a corresponding *Onward Tree Subsequential Transducer* (OTST). Fig.1 (left and center) illustrates these concepts (and construction), which are the basis of the so-called *Onward Subsequential Transducer Inference Algorithm* (OSTIA), by Oncina [14, 15].

Given an input-output training sample T, the OSTI Algorithm works by *merging states* in the $OTST(T)$ as follows [15]: All pairs of states of $OTST(T)$ are orderly considered level by level, starting at the root, and, for each of these pairs, the states are tentatively merged. If this results in a non-deterministic state, then an attempt is made to restore determinism by recursively pushing-back some output substrings towards the leaves of the transducer (i.e., partially undoing the onward construction), while performing the necessary additional state merge operations. If the resulting transducer is subsequential, then (all) the merging(s) is (are) accepted; otherwise, a next pair of states is considered in the previous transducer. A transducer produced by this procedure from the OTST of Fig.1 (center) is shown in Fig.1 (right). Note that this resulting transducer is consistent with *all* the training pairs in T and makes a suitable generallization thereof.

All these operations can be very efficiently implemented, yielding an extremely fast algorithm that can easily handle huge sets of training data. It has formally been shown that OSTIA always converges to any target subsequential transduction for a sufficiently large number of training pairs of this transduction [15].

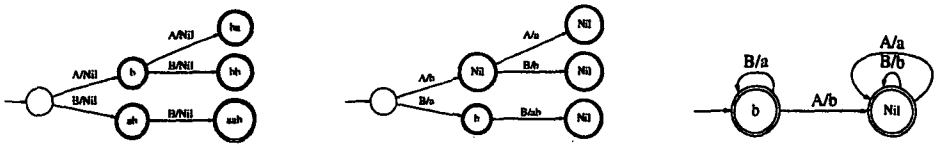

Figure 1. Learning a Subsequential Transducer from the input-output sample T={(A,b), (B,ab), (AA,ba), (AB,bb), (BB,aab)}. *Left*: Tree Subsequential Transducer $TST(T)$; *Center*: Onward Tree Subsequential Transducer $OTST(T)$; *Right*: transducer yield by OSTIA. Each state contains the output string that the function σ associates to this state.

The learning strategy followed by OSTIA tries to generalize the training pairs as much as possible. This often leads to very compact transducers that accurately translate *correct* input text. However, this compactness often entails excessive *over-generalization* of the input and output languages, allowing nearly meaningless input sentences to be accepted, and translated into even more meaningless output! While this is not actually a problem for perfectly *correct text* input, it leads to dramatic failures when dealing with not exactly correct text or (even "correct") *speech* input.

A possible way to overcome this problem is to limit generalization by imposing adequate Language Model (LM) constraints: the learned SSTs should *not* accept input sentences or produce output sentences which are not consistent with given LMs of the input and output

languages. These LMs are also known as *Domain* and *Range* models [17]. Learning with Domain and/or Range constraints can be carried out with a version of OSTIA called OSTIA-DR [16, 17]. This version was used in the work presented in this paper.

Subsequential Transducers and the OSTI (or OSTI-DR) Algorithm have been very successfully applied to learning several quite contrived (artificial) translation tasks [15]. Also, it has recently been applied to Language Translation [25, 9, 1] and Language Understanding, as will be discussed here below. Among many possibilities for (finite-state) modeling the input and output languages, here we have adopted the well-known bigrams [8], which can be easily learned from the same (input and output) training sentences used for OSTIA-DR.

3 Reducing the demand for training data

The amount of training data required by OSTIA(-DR)–learning is directly related with the size of the vocabularies and the amount of input-output *asynchrony* of the translation task considered. This is due to the need of "delaying" the output until enough input has been seen. In the worst case, the number of states required by a SST to achieve this delaying mechanism can grow as much as $O(n^k)$, where n is the number of (functionally equivalent) words and k the length of the delay.

Techniques to reduce the impact of k were studied in [29]. The proposed methods rely on *reordering* the words of the (training) output sentences on the base of *partial alignments* obtained by statistical translation methods [5]. Obviously, adequate mechanisms are provided to recover the correct word order for the translation of new test input sentences [29].

3.1 Using word/phrase categorization

On the other hand, techniques to cut down the impact of vocabulary size were studied in [28]. The basic idea was to substitute words or groups of words by labels representing their syntactic (or semantic) *category* within a limited rank of options. Learning was thus carried out with the categorized sentences, which involved a (much) smaller effective vocabulary. The steps followed for introducing categories in the learning and transducing processes began with category identification and categorization of the corpus. Once the categorized corpus was available, it was used for training a model: the *base transducer*. Also, for each category, a simple transducer was built: its *category transducer*. Finally, category expansion was needed for obtaining the final sentence-transducer: the arcs in the base transducer corresponding to the different categories were expanded using their category transducers.

Note that, while all the transducers learned by OSTIA-DR are subsequential and therefore deterministic, this embedding of categories generally results in final transducers that are no longer subsequential and often they can be ambiguous. Consequently, translation can not be performed through deterministic parsing and Viterbi-like Dynamic Programming is required.

Obviously, categorization has to be done for input/output *paired clusters*; therefore adequate techniques are needed to represent the actual identity of input *and* output words in the clusters and to recover this identity when parsing test input sentences. This recovering is made by keeping referencies between category labels and then solving them with a postprocess filter. This method is explained in detail in [1]. Text-input experiments using these techniques were presented in [28]. While the direct approach degrades rapidly with increasing vocabulary sizes, categorization keeps the accuracy essentially unchanged.

3.2 Coping with undertraining through Error Correcting

The performance achieved by a SST model (and for many other types of models whatsoever) tends to be poor if the input sentences do not strictly comply with the syntactic restrictions imposed by the model. This is the case of syntactically incorrect sentences, or correct sentences whose precise "structure" has not been exactly captured because it was not present in the training data.

Both of these problems can be approached by means of Error-Correcting Decoding (ECD) [3, 29]. Under this approach, the input sentence, x, is considered as a *corrupted* version of some sentence, $\hat{x} \in L$, where L is the domain or input language of the SST. The corruption process is modeled by means of an Error Model that accounts for insertion, substitution and deletion *"edit errors"*. In practice, these "errors" should account for likely vocabulary variations, word disappearances, superfluous words, repetitions, and so on. Recognition can then be seen as an ECD process: given x, find a sentence $\hat{x}$ in L such that the distance form $\hat{x}$ to x, measured in terms of edit operations (insertions, deletions and substitutions) is minimum[2].

Given the finite-state nature of SST Models, Error Models can be *tightly* integrated, and combined error-correcting decoding and translation can be performed very efficiently using fast ECD beam-search, Viterbi-based techniques such as those proposed in [3].

4 Experiments

The chosen task in our experiments was the translation from Spanish sentences specifying times and dates into sentences of a formal semantic language. This is in fact an important subtask that is common to many real-world LU applications of much interest to industry and society. Examples of this kind of applications are flight, train or hotel reservations, appointment schedules, etc. [7, 11, 12]. Therefore, having an adequate solution to this subtask can significantly simplify the building of successful systems for these applications (another work on this subtask can be found in [6]).

The chosen formal language has been the one used in UNIX command "at". This simple language allows both absolute and relative descriptions of time. From these descriptions, the "at" interpreter can be directly used to obtain date/time interpretations in the desired format. The correct syntax of "at" commands is described in the standard Unix documentation (see, e.g. [30]). Fig. 2 shows some training pairs that have been selected from the training material.

Starting from the given context-free-style syntax description of the "at" command [30], and knowledge-based patterns of typical ways of expressing dates and times in natural, spontaneous Spanish, a large corpus of pairs of "natural-language"/at-language sentences has been artificially constructed. This is intended to be the first step in a bootstrapping development. On-going work on this task is aimed at (semi-automatically) obtaining additional corpora produced by native speakers. The corpus generation procedure incorporated certain "category labels", such as hour, month, day of week, etc. We have used a similar process for defining and generating subcorpora in which every input and its corresponding semantic coding belong to the different categories. We finally have obtained an uncategorized version of the categorized corpus, by means of randomly instantiating the category marks in the samples. The examples found on figure 2 come from this uncategorized corpus, while figure 3 shows the corresponding categorized pairs.

[2] Note that while only simple deterministic ECD is considered in this paper, ECD can be easily formulated in a more powerful, *stochastic* manner [2].

("dos minutos después de la una y media", 01 : 30 + 2 MINUTE) *(two minutes after one thirty)*
("dentro de una hora", NOW + 1 HOUR) *(in one hour)*
("el martes, a la hora del té, mas un minuto", TEATIME TUE + 1 MINUTE) *(on thursday, at teatime plus one minute)*
("el catorce de octubre del año dos mil tres, a las diecisiete horas y cinco minutos", 17 : 05 OCT 14 , 2003) *(on october the first, year two thousand and three, at seventeen hours and five minutes)*

Figure 2. Sample of selected training pairs for the date specification task.

("**inc-number** minutos después de **h24 mm**", **h24 : mm + inc-number** MINUTE)
("dentro de una hora", NOW + 1 HOUR)
("el **day-of-week** , a **t-dest** , mas un minuto", **t-dest day-of-week** + 1 MINUTE)
("el **day-txt** de **month-name** del año **year-name**, a **h24 mm**", **h24 : mm month-name day-txt , year-name**)

Figure 3. Sample of categorized pairs for the date specification task.

We have generated a training corpus of 48353 different, uncategorized translation pairs, and a *disjoint* test set with 1331 translation pairs. We have presented the OSTIA-DR with 8 training subsets of sizes increasing from 1817 up to 48353. We also have presented OSTIA-DR with the same, but categorized, training subsets. In this case, the number of different pairs went from 1384 up to 12381. Figure 4 shows the size of categorized corpora vs. uncategorized corpora. The input language vocabulary has 108 words, and the output language has 125 semantic symbols. We have used 11 different category labels.

In the categorized experiments, a sentence-transducer was inferred from the categorized sentences, and a (small) category-transducer for each one of the categories. The final transducer, which is able to translate noncategorized sentences, was build up by the embedding of the category-transducers into the sentence-transducers. The output yielded by this final transducer includes category labels and their corresponding instances, as found in the translation process. The definitive translations of the test set inputs are obtained by means of a simple filter that resolves the dependencies. The sizes of the inferred transducers are shown on figure 5.

Performance has been measured in terms of both semantic-symbol error and full-sentence matching rates. The translation of the test set inputs has been computed using both the standard Viterbi algorithm and the Error Correction techniques, outlined on sections 3.1 and 3.2. The results are shown in figure 6.

A big difference in performance between the uncategorized and categorized training procedures can be observed. Semantic-symbol error rates are much lower in the categorized experiments than in the uncategorized ones. We can also appreciate a remarkable decrease in semantic-symbol error rates of Error Correcting with respect to Viterbi translations, specially for smaller training corpus. The full-sentence matching rate also exhibited a strong improve-

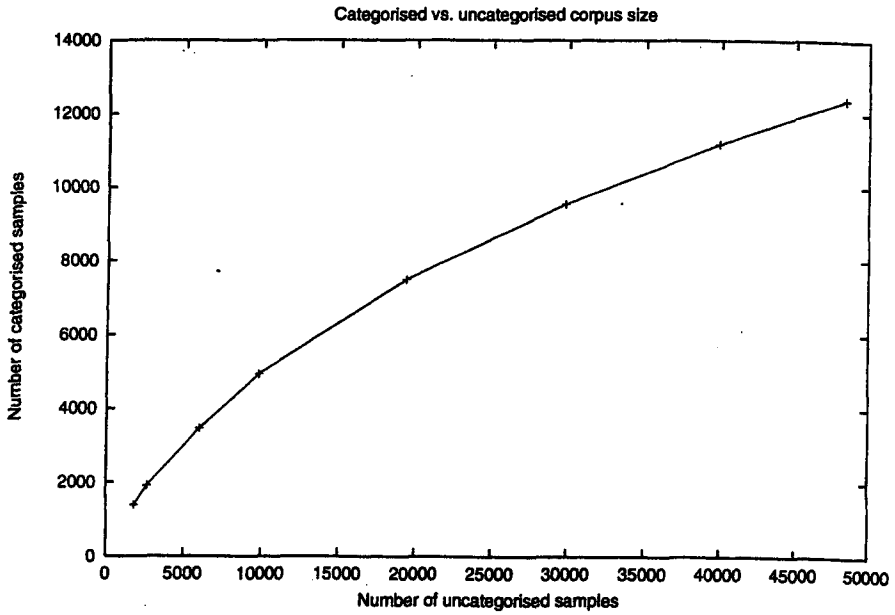

Figure 4. Corpora size before and after categorization.

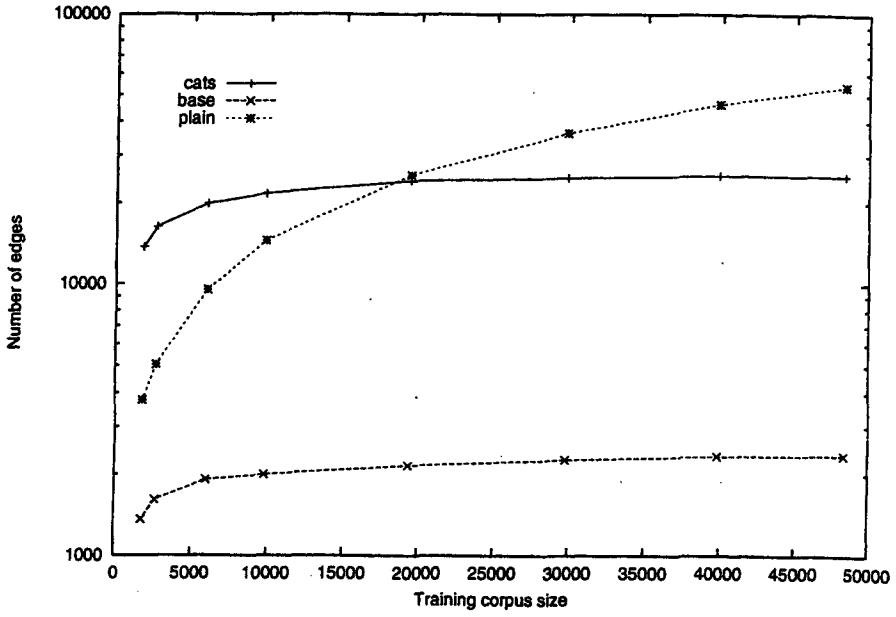

Figure 5. Inferred transducers sizes. The size is expressed in number of edges: "base" stands for the transducer containing category labels, while "cats" stands for the final sentence-transducer which is calculated by embedding the (small) category-transducers into the "base" one; "plain" stands for the uncategorized sentence-transducer.

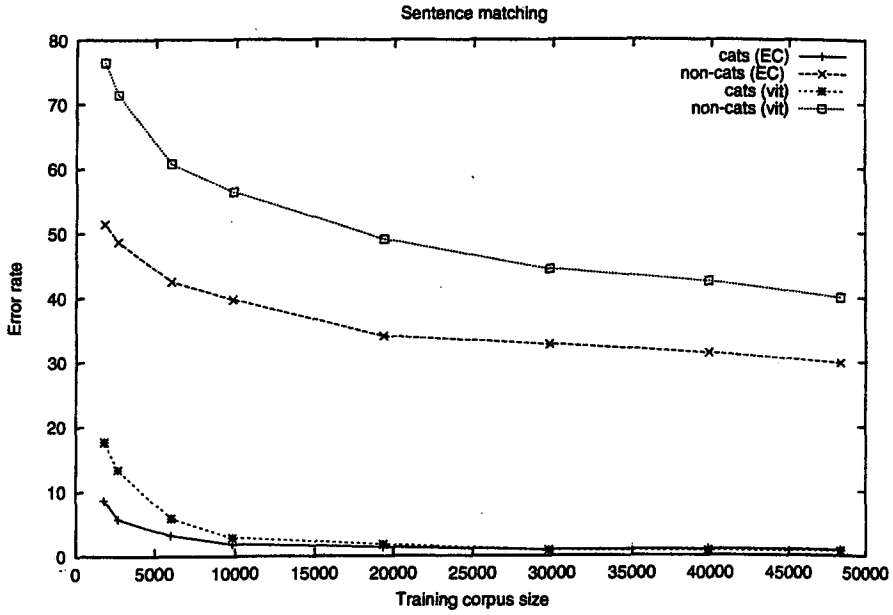

Figure 6. Semantic-symbol error rates. On the legend, "cats" stands for the categorised experiments, and "non-cats" for the non-categorized ones. Transductions in "EC" have been computed using Error Correcting techniques, and in "vit" using the standard Viterbi algorithm.

ment by using categorization: while uncategorized training only achieves 30%-40% matching rate, the categorized one yields up to 98%.

5 Conclusions

In this work, we have presented some successful experiments on a non-trivial, useful task in natural language understanding. Finite-State models have been learnt by the OSTIA-DR algorithm. Our attention has been centered in the possibility of reducing the demand for training data by categorizing the corpus. The experiments show a very big difference in performance between the categorized and plain training procedures. In this task, we only obtain useful results if we use categories.

The Error Correcting technique for translation also permits reducing the size of corpora and still obtain useful error rates. In our task, we got a 3% in semantic-symbol error rate for a training set of approximately 6000 pairs, while for the same level of performance using the standard Viterbi algorithm requires some 10000 training pairs. This 3% error rate result corresponds to a full-sentence matching rate of 90%.

On-going work on these techniques is aimed at obtaining additional training data by native speakers, so as to improve the system by following a *bootstrapping* procedure: the system will be trained on this additional natural or spontaneous data, the acquisition of which is driven by the system itself, guided by given task-relevant semantic stimuli. This process can be repeated until the resulting system exhibits a satisfactory performance. On the other hand,

transducers generated by the embedding procedure described in this paper may turn out to be ambiguous. Work is also being done on applying stochastical extensions of transducers, so as to deal with ambiguities by reflecting the appearance probability distribution of sentences in the training corpus. These distributions are being estimated by Maximum-Likelihood, Conditional Maximum-Likelihood, or Maximum Mutual Information Estimation [18]. The results of this work will be useful as a subtask of the so-called "Tourist Task", which is a hotel reservations task introduced in the EuTrans project.[1, 25]

References

1. J.C AMENGUAL, J.B.BENEDÍ, F.CASACUBERTA, A. CASTAÑO, A. CASTELLANOS, D. LLORENS, A. MARZAL, F. PRAT, E. VIDAL AND J.M.VILAR: "Using Categories in the Eutrans System". *ACL-ELSNET Workshop on Spoken Language Translation*, Madrid, Spain, pp. 44-52. (1997)
2. J.C. AMENGUAL, E. VIDAL. *Two Different Approaches for Cost-efficient Viterbi Parsing with Error Correction*. Proc. of the SSPR'96, IAPR International Workshop on Structural and Syntactical Pattern Recognition, August 20–23, 1996, Leipzig. To be published in the Proceedings.
3. J.C. AMENGUAL, E. VIDAL AND J.M. BENEDÍ. "Simplifying Language through Error-Correcting Decoding". *Proceedings of the ICSLP96 (IV International Conference on Spoken Language Processing)*. To be published. October, 1996.
4. J. BERSTEL. *Transductions and Context-Free Languages*. Teubner, Stuttgart. 1979.
5. P.F.BROWN ET AL.. "A Statistical Approach to Machine Translation". *Computational Linguistics*, Vol. 16, No.2, pp.79-85, 1990.
6. J.G.BAUER, H.STAHL, J.MLLER: "A One-pass Search Algorithm for Understanding Natural Spoken Time Utterances by Stochastic Models". Proc. of the EUROSPEECH'95, Madrid, Spain, vol.I, pp. 567-570. (1995)
7. C.T.HEMPHILL, J.J.GODFREY, G.R.DODDINGTON. "The ATIS Spoken Language Systems, pilot Corpus". *Proc. of 3rd DARPA Workshop on Speech and Natural Language*, pp. 102-108, Hidden Valley (PA), June 1990.
8. F. JELINEK: "Language Modeling for Speech Recognition". In [10] (1996).
9. V.JIMENEZ, A.CASTELLANOS, E.VIDAL. "Some results with a trainable speech translation and understanding system". In *Proceedings of the ICASSP-95*, Detroit, MI (USA), 1995
10. A.KORNAI (ED.); *Proceedings of the ECAI'96 Workshop: Extended Finite State Models of Language*. Budapest, 1996.
11. A.LAVIE, A.WAIBEL, L.LEVIN, M.FINKE, D.GATES, M.GAVALDÀ, T.ZEPPENFELD AND P.ZHAN: "JANUS-III: Speech-to-speech Translation in Multiple Languages", Proc. of the ICASSP'97, Munich, Germany, vol. I, pp. 99-102. (1997)
12. E. MAIER AND S. MCGLASHAN: "Semantic and Dialogue Processing in the VERBMOBIL Spoken Dialogue Translation System", In *Proceedings in Artificial Intelligence: CRIM/FORWISS Workshop on Progress and Prospects of Speech Research and Technology*, H. Niemann, R. de Mori and G. Hanrieder (eds.), Infix, pp. 270–273. (1994)
13. M.MOHRI. "Finite-State Transducers in Language and Speech Processing". Computational Linguistics 23:2, 269-311.
14. J.ONCINA. "Aprendizaje de Lenguages Regulares y Funciones Subsecuenciales". Ph.D. diss., Universidad Politecnica de Valencia, 1991.
15. J.ONCINA, P.GARCIA, E.VIDAL. "Learning Subsequential Transducers for Pattern Recognition Interpretation Tasks". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.15, No.5, pp.448-458. May, 1993.
16. J.ONCINA, A.CASTELLANOS, E.VIDAL, V.JIMENEZ. "Corpus-Based Machine Translation through Subsequential Transducers". *Third Int. Conf. on the Cognitive Science of Natural Language Processing*, proc., Dublin, 1994
17. J.ONCINA, M.A.VAR. "Using domain information during the learning of a subsequential transducer". In Laurent Miclet and Colin de la Higuera, editors, *Grammatical Inference: Learning Syntax from Sentences, Lecture Notes in Computer Science*, vol. 1147, pp. 301-312. Springer-Verlag. 1996

18. D. PICÓ. "Algorismes d'aprenentatge per a traductors regulars estocàstics amb λ-regles". Master's Thesis. Universitat Politècnica de València. València. Spain. 1997.
19. R. PIERACCINI, E. LEVIN. "Stochastic Representation of Semantic Structure for Speech Understanding". EUROSPEECH'91, Proc., Vol. 2, pp.383-386. Genoa Sept, 1991.
20. R. PIERACCINI, E. LEVIN, E. VIDAL. "Learning How To Understand Language". EUROSPEECH'93, proc., Vol.2, pp. 1407-1412. Berlin, Sept, 1993.
21. N.PRIETO, E.VIDAL. "Learning Language Models through the ECGI method". Speech Communication, No.11, pp.299-309. 1992.
22. C. REUTENAUER. "Subsequential functions: characterization, minimization, examples". In J. Kelemen, editor. *Proceedings of the International Meeting of Young Computer Scientists, Lecture Notes in Computer Science*, vol. 464, pp. 62-79. Springer-Verlag, 1990.
23. K.SEYMORE, R.ROSENFELD. "Scalable Backoff Language Models". *ICSLP-96, proc.*. pp.232-235. Philadelfia, 1996.
24. E. VIDAL: "Language Learning, Understanding and Translation", In *Proceedings in Artificial Intelligence: CRIM/FORWISS Workshop on Progress and Prospects of Speech Research and Technology*, H. Niemann, R. de Mori and G. Hanrieder (eds.), pp. 131–140. Infix, (1994).
25. E. VIDAL: "Finite-State Speech-to-speech Translation", Proc. of the ICASSP'97, Munich, Germany, vol.I, pp. 111-122. (1997)
26. E.VIDAL, F.CASACUBERTA, P.GARCIA. "Grammatical Inference and Automatic Speech Recognition". In *Speech Recognition and Coding. New Advances and Trends*, J.Rubio and J.M.Lopez, Eds. Springer Verlag, 1994.
27. E.VIDAL, D.LLORENS. "Using knowledge to improve N-Gram Language Modeling through the MGGI methodology". In *Grammatical Inference: Learning Syntax from Sentences*, L.Miclet, C.De La Higuera, Eds. LNAI (1147), Springer-Verlag, 1996.
28. J.M. VILAR, A. MARZAL, E. VIDAL: "Learning Language Translation in Limited Domains using Finite-State Models: some Extensions and Improvements". *Proceedings of the EUROSPEECH-95*, Madrid, Spain, pp. 1231-1234. (1995)
29. J.M. VILAR, E. VIDAL AND J.C. AMENGUAL: "Learning Extended Finite State Models for Language Translation". *Proceedings of the ECAI96* (12th European Conference on Artificial Intelligence). August (1996).
30. Linux system documentacion, at directory "/usr/doc/at" (Debian distribution). Also, see "man at" on a Unix system.