Machine Learning, 44, 143–159, 2001

Language Simplification through Error-Correcting and Grammatical Inference Techniques

JUAN-CARLOS AMENGUAL jcamen@inf.uji.es
Universidad Jaume I, Campus de Riu Sec, 12071 Castellón, Spain

ALBERTO SANCHIS asanchis@iti.upv.es
ENRIQUE VIDAL evidal@iti.upv.es
JOSÉ-MIGUEL BENEDÍ jbenedi@iti.upv.es
Instituto Tecnológico de Informática, Camino de Vera s/n, 46071 Valencia, Spain

Editors: Colin de la Higuera and Vasant Honavar

Abstract. In many language processing tasks, most of the sentences generally convey rather simple meanings. Moreover, these tasks have a limited semantic domain that can be properly covered with a simple lexicon and a restricted syntax. Nevertheless, casual users are by no means expected to comply with any kind of formal syntactic restrictions due to the inherent "spontaneous" nature of human language. In this work, the use of error-correcting-based learning techniques is proposed to cope with the complex syntactic variability which is generally exhibited by natural language. In our approach, a complex task is modeled in terms of a basic finite state model, F, and a stochastic error model, E. F should account for the basic (syntactic) structures underlying this task, which would convey the meaning. E should account for general vocabulary variations, word disappearance, superfluous words, and so on. Each "natural" user sentence is thus considered as a corrupted version (according to E) of some "simple" sentence of $L(F)$. Adequate bootstrapping procedures are presented that incrementally improve the "structure" of F while estimating the probabilities for the operations of E. These techniques have been applied to a practical task of moderately high syntactic variability, and the results which show the potential of the proposed approach are presented.

Keywords: language processing, error-correcting techniques, bootstrapping, incremental learning algorithms, edit operations

1. Introduction

Among many application fields of *Grammatical Inference* (GI), perhaps one that is nowadays considered the most promising is *Speech and Natural Language Processing* (SNLP) (Vidal, Casacuberta, & García, 1995). However, currently available GI technology can only be considered adequate for language involving simple, regular and clean sentences, but is clearly not mature enough to truly model natural and spontaneous language involving contrived syntactic constructions, large vocabularies, etc.

Fortunately, many interesting SNLP applications deal with *Limited Domains* (LD). This means that the universe of discourse is bounded, mainly in terms of the semantic scope involved. Although the language generally used in an LD task may be *quite complex*, having large vocabularies and many irregularities, the underlying syntax and lexicon actually needed to convey the required meaning may *not be so complex*.

The proposed framework to cope with this situation is *Error-Correcting* (EC): the simpler underlying "clean" language is modeled by a basic, canonical model F and spontaneous language effects (i.e. vocabulary variations, word disappearance, superfluous words, etc.) are modeled by means of an adequate *error model*, E. We have named this approach *Language Simplification* (LS).

The canonical model could be a finite state model (FSM) $F = (Q, \Sigma, E, q_0, Z)$, where Q is the set of states, Σ is an alphabet which represents the basic lexicon, $E \subseteq Q \times \Sigma \times Q$ is the set of transitions which represents the canonical syntactic structures required to convey the semantics of the task, q_0 is the initial state, and Z is the set of final states. Additionally, a function $\phi : Q \times \Sigma \times Q \rightarrow [0..1]$ can be used to define a probability distribution over the strings accepted by F. On the other hand, the error model E accounts for insertions, substitutions, and deletions of symbols belonging to Σ (Fu, 1982; Gonzalez & Thomason, 1978).

Under this framework, a given natural sentence, $\bar{x}$, is seen as a "distorted" version of some "clean" sentence, $\bar{y}$, supposedly belonging to the (stochastic) language of F, $L(F)$. On the other hand, the distortion process leading from $\bar{y}$ to $\bar{x}$ is assumed to be driven by E. Given $\bar{x}$, F, and E, (stochastic) *Error-Correcting Parsing* (ECP) can thus be used to uncover which $\bar{y}$ was the most likely source from which the observed sentence $\bar{x}$ was produced.

For example, let us use an application task where users make queries to a Spanish geographical database (see Section 5) employing *unconstrained* natural language (text or speech). In addition, assume that the language model for this task only accepts the canonical syntactic structure "*How many rivers flow into the SEA sea?*"[1] (sentence $\bar{y}$) for *referring* to the semantic concept *the number of rivers that flow into the SEA sea* (which, in turn, can be easily transformed into a database query). Then, a user makes the following *natural* query: "*What is the number of rivers that flow into the SEA sea?*" (sentence $\bar{x}$). Following the EC framework in this example, the desirable output, the "clean sentence", would be $\bar{y}$ after parsing the "distorted" input sentence $\bar{x}$ (see also subsection 5.1). A possible sequence of EC operations leading from $\bar{y}$ to $\bar{x}$ would be the substitution of "*How*" by "*What*", the insertions of "*is*" and "*the*", the substitution of "*many*" by "*number*", the insertions of "*of*" and "*that*", and the substitutions of "*rivers*", "*flow*", "*into*", "*the*", "*SEA*", and "*sea?*" for themselves.

The following section describes the basics of EC, while subsection 2.1 specifically addresses the problem of the estimation of the error model probabilities when using stochastic ECP, which is important to achieving our goal. This brings about the need for developing adequate learning procedures from training data, which take the *key role* played by "clean" sentences into account. They not only convey the semantics of the task but also *bias* the estimation for the probabilities of EC operations. Since they are needed to both adequately estimate the probabilities of E and build the canonical FSM, F, our approach raises an important problem: "clean" sentences are "*hidden*", in the sense that they cannot be directly observed from the natural performance of the users in the application considered. A solution can be achieved through the *bootstrapping* approach proposed in Section 3, (initially at least) resorting to human "expert" assistance. One of the tasks of the human "expert" would be, for instance, to build a first canonical FSM.

Realistically, the canonical FSM, F, cannot be expected to provide a complete coverage of the *natural* language that would cope with the entire semantic/pragmatic scope of the task

under consideration. Even when using EC techniques, there can exist sentences which are very difficult or impossible to parse due to: i) syntactic variations regarding $L(F)$ that cannot be properly supported by an ins-sub-del error model; and ii) the fact that their meaning is not covered by any syntactic structure modeled by F. In both cases, these sentences can be used to "refine" F. Obviously, one can repeatedly resort to human expert help for these refinements, but this becomes expensive or even impractical once the complexity of F goes beyond a certain level: it becomes increasingly difficult to assure the consistence and appropriateness of the expert's updates with those made in previous steps.

An alternative, more interesting approach is to perform these "refinements" by means of GI techniques. In this work, we propose a heuristic approach to inductive "FSM refinement" which is specifically adapted to the general EC framework adopted. This approach is reminiscent of the ideas underlying the "Error-Correcting Grammatical Inference" algorithm (ECGI), which were introduced by Rulot and Vidal (1987, 1988). A recently developed efficient ECP algorithm (Amengual & Vidal, 1998) overcomes the original ECGI limitation of having to work only with *acyclic* FSMs. On the other hand, the new ECGI-like algorithm includes new heuristics that have specifically been designed to cope with some constructions which often appear in natural language applications. This algorithm is described in Section 4. Section 6 presents experiments on a Language Understanding task involving natural language queries to a Geographic Information database as described in Section 5. Section 7 presents the conclusions that have been drawn.

2. Basic concepts about error correction

A classical *error model*, E, comprising insertions, substitutions and deletions of *symbols* belonging to some alphabet Σ has been used in this work. Formally speaking, an error operation is a pair (a, b) of symbols belonging to $\Sigma \cup \{\backslash\}$, with $\backslash$ being the *null* or *empty* symbol.[2] In general, a pair (a, b) means that symbol a is changed for symbol b. Thus, if we have a string $\bar{y}$ of the form $\bar{u}a\bar{w}$, this string can be *transformed* into the string $\bar{x}$ of the form $\bar{u}b\bar{w}$ by applying the error operation (a, b). Following this definition, error operations depend only on the pair of symbols involved in the transformation, independently of the position in the string where the error operation has been applied. Therefore, the whole set of error model operations can be classified into these three categories:

- Substitution operations. All possible pairs (a, b) where $a \neq \backslash$ and $b \neq \backslash$. These include the particular case $a = b$, i.e. non-error operations.
- Deletion operations. All possible pairs (a, b) where $a \neq \backslash$ and $b = \backslash$.
- Insertion operations. All possible pairs (a, b) where $a = \backslash$ and $b \neq \backslash$.

Henceforth, the set of possible error model operations will be considered as *E operations*. Each E operation also has a *weight* given by the function $\rho : (\Sigma \cup \{\backslash\}) \times (\Sigma \cup \{\backslash\}) \rightarrow \Re^+$. The weights of E operations have been heuristically set in many applications (Sankoff & Kruskal, 1983), (perhaps) with the Levenshtein edit distance (Kruskal, 1983) being the most widely used. This distance is based on defining $\rho(a, a) = 0, \rho(a, b) = \rho(a, \backslash) = \rho(\backslash, b) = 1, \ \forall a, b \in \Sigma \ a \neq b$.

It is assumed that every possible string $\bar{y}$ is input to a *noisy* channel C. The output of C is the string $\bar{x}$, which is likely to be different from $\bar{y}$. C works by performing 0, 1, or more *consecutive* transformations over $\bar{y}$ using the set of E operations, whose weight is given by ρ. Therefore, if $\mathcal{S}$ is the set of different sequences of error operations which are able to transform the original string $\bar{y}$ into $\bar{x}$, a *dissimilarity* function between two strings, $\mathcal{D} : \Sigma^* \times \Sigma^* \to \Re^+$ is computed as $\mathcal{D}(\bar{x}, \bar{y}) = \min_{O \in \mathcal{S}} \Upsilon(O)$, where Υ yields the *cost* of a sequence of error operations, O, and is usually computed as:

$$\Upsilon(O) = \sum_{\forall (a,b) \in O} \rho(a, b)$$

E can be easily integrated with a FSM, F, in general, producing a *non-deterministic* expanded FSM by adding the error *transitions* as follows:

$$\begin{aligned}
&\forall q \in Q, \forall a \in \Sigma : \delta(q, a) = \mathcal{A}, \mathcal{A} \in 2^Q, \\
&\textit{Substitutions} \quad \delta(q, b) = \mathcal{A}, \forall b \in \Sigma, b \neq a \\
&\textit{Deletions} \quad \delta(q, \backslash) = \mathcal{A} \\
&\textit{Insertions} \quad \delta(q, c) = \{q\}, \forall c \in \Sigma
\end{aligned}$$

Each of these transitions has a weight which is associated to the corresponding error model operation, as given by ρ. Note that these transitions do not have to be explicitly represented in the expanded FSM. Instead they will be *dynamically* expanded during the parsing process, taking only the input symbol to be parsed into account (Amengual & Vidal, 1998).

Given a supposedly distorted string $\bar{x}$, as well as a FSM F (that accepts the input language of the noisy channel C), and the set of values of the function ρ then a *minimum-distance* error-correcting parser is an algorithm that solves the following search problem (Fu, 1982):

$$\bar{y}^* = \operatorname*{argmin}_{\bar{y} \in L(F)} \mathcal{D}(\bar{x}, \bar{y}) \tag{1}$$

An important issue when using EC techniques is the choice (or "learning") of an adequate error-weight function ρ, given the strong sensitivity to changes in the operation costs shown by these techniques. Some relevant works have been done in this respect (Bunke & Csirik, 1995; Rice, Bunke, & Nartker, 1997). Perhaps the most interesting weighting scheme is the one based on stochastic assumptions (Amengual, 1999; Amengual & Vidal, 2000; Ristad & Yianilos, 1998).

For a probabilistic definition of E, we only need to re-define the function ρ as $\rho : (\Sigma \cup \{\backslash\}) \times (\Sigma \cup \{\backslash\}) \to [0..1] \wedge \sum_{\forall b \in \Sigma \cup \{\backslash\}} \rho(a, b) = 1 \forall a \in \Sigma \cup \{\backslash\}$ (Fu, 1982). The function Υ is now defined as the probability of a sequence of error operations:

$$\Upsilon(O) = \prod_{\forall (a,b) \in O} \rho(a, b)$$

Therefore, the probability for E to produce $\bar{x}$ from $\bar{y}$ is:

$$P_E(\bar{x} \mid \bar{y}) = \sum_{\forall O \in \mathcal{S}} \Upsilon(O)$$

From this point of view, we are interested in a string of $L(F)$ that is most likely transformed by C into $\bar{x}$. A *maximum likelihood* error-correcting parser is an algorithm that yields a solution by solving the following search problem (Fu, 1982):

$$\bar{y}^* = \underset{\bar{y} \in L(F)}{\operatorname{argmax}} P(\bar{y} \mid \bar{x}) \tag{2}$$

Equivalently, by applying the Bayes rule, we have:

$$\bar{y}^* = \underset{\bar{y} \in L(F)}{\operatorname{argmax}} P_F(\bar{y}) \cdot P_E(\bar{x} \mid \bar{y}) \tag{3}$$

where $P_F(\bar{y})$ is the probability for F to accept $\bar{y}$, and, by using the *maximum* approximation for both $P_F(\bar{y})$ and $P_E(\bar{x} \mid \bar{y})$, this finally results in:

$$\bar{y}^* = \underset{\substack{\bar{y} \in L(F) \\ O \in \mathcal{S}}}{\operatorname{argmax}} P_F(\bar{y}) \cdot \Upsilon(O) \tag{4}$$

This approximation has the advantage that its solution yields not only the most likely "clean" string in $L(F)$, $\bar{y}^*$, but also the most likely error transformation of $\bar{y}^*$ into $\bar{x}$ (something that could not be obtained with a pure "Forward-like" approach, which on the other hand represents much higher computational costs). As will be shown later in Section 4, the individual errors of this transformation will be used as the basis for learning successive refinements of F.

An *efficient* Viterbi-like search algorithm which solves the problem posed by equations (1) and (3) (under the *max* approximation) has recently been developed (Amengual & Vidal, 1998). Full algorithmic details are given in that paper. Another efficient algorithm which is capable of solving the same search problem was proposed in Gregor and Harris (1995), but, unfortunately, this solution is restricted to acyclic or "self-looped" (left-to-right) FSMs.

Let us now focus on the estimation of the parameters of F and E following equation (3), where *two different* probability distributions are involved. On the one hand, P_F can be independently learned using a training corpus composed of strings that are accepted by the original FSM, F, by applying maximum likelihood techniques such as those proposed in Casacuberta (1996). On the other hand, P_E also has to be learned from training data.

Only recent works seem to propose adequate solutions to this problem (Amengual, 1999; Amengual & Vidal, 2000; Ristad & Yianilos, 1998). The most interesting of these (keeping the simplification approach in mind) are those based on the well-known Expectation-Maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977). An important reason for following this approach is the expected data sparseness: the EM algorithm can pull all of the relevant (probabilistic) information out of training data, even if this data is scarce.

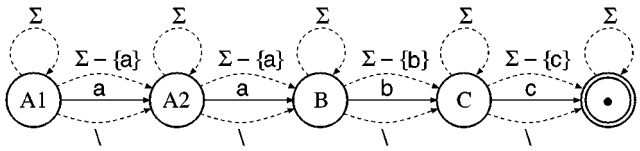

Figure 1. A finite state automaton, representing the sentence *aabc*, extended to include E, which comprises insertions (transitions labeled Σ), substitutions (transitions labeled $\Sigma - \{\text{the correct transition symbol}\}$) and deletions (transitions labeled \).

2.1. Estimation of error model parameters

Direct maximum likelihood estimation of E operations requires a training corpus, $\mathcal{T}$, consisting of pairs of ("distorted"—"clean") sentences. Each "clean" sentence is represented as a Markov Model as shown in figure 1 (Amengual, 1999; Amengual & Vidal, 2000).

$$V = P_E(\mathcal{T}) = \prod_{\forall(\bar{x},\bar{y})\in\mathcal{T}} P_E(\bar{x} \mid \bar{y}) \tag{5}$$

In order to maximize V, the well-known Baum-Eagon inequality (Baum & Eagon, 1967) can be used to iteratively compute new estimates for the probabilities of E operations. This leads to an EM-like re-estimation procedure which is very close to the well-known Forward-Backward algorithm, which is generally used to estimate the parameters of Hidden Markov Models in speech recognition (Casacuberta, 1996). Full details of this iterative re-estimation procedure are given in Amengual (1999) and Amengual and Vidal (2000), where the small differences between our approach and that presented in Ristad and Yianilos (1998) are explained.

This re-estimation scheme can often be improved in practice by a few iterations of "final Viterbi re-estimation" after EM convergence, taking the estimates yielded by the Forward-Backward-like procedure as the *initial starting values*. This *strengthens* the highest probabilities achieved by EM (which are often too "flat"), thus *improving* the estimates of E operations. The Viterbi approach amounts to changing the optimization criterion (5) by:

$$V = \hat{P}_E(\mathcal{T}) = \prod_{\forall(\bar{x},\bar{y})\in\mathcal{T}} \hat{P}_E(\bar{x} \mid \bar{y}) \tag{6}$$

where $\hat{P}_E(\bar{x} \mid \bar{y})$ is the probability of the *best* alignment or "Viterbi approximation" to $P_E(\bar{x} \mid \bar{y})$, which can be computed as discussed in the previous section (see Eqs. (3) and (4)).

Even if error-correcting techniques are used, it is likely that some sentences in Σ^* be rejected. This can be produced if some operations of the error model have not been trained, i.e. their probability is null. To solve this problem, we have used five *constants* to smooth the relative frequencies of use of error operations (Amengual & Vidal, 2000): $\epsilon_{ins}, \epsilon_{del}, \epsilon_{sub}$, and ϵ_{noe} (to be added to the counts of the insertion, deletion, and substitution errors, and

substitutions of symbols for themselves, respectively). A last constant, ϵ_{unk}, is the value for the counts associated to all possible errors involving the insertion of or the substitution by the *unknown* word. In this way, out-of-vocabulary words can be coped with. In the experiments presented in Section 6, their values have been set to 1.

3. Progressive parameter estimation through bootstrapping techniques

As already introduced in Section 1, our LS paradigm raises the problem that no "clean" or "simplified" sentences are naturally available to train the required error model. Sentences of this kind are really *needed* to adequately estimate E operations, since they "*guide*" the training procedure outlined in Section 2.1 towards accounting for the necessary vocabulary variations that would eventually allow the achievement of the right[3]"simplified" sentences. Moreover, "clean" sentences are required to build F, the canonical FSM. A solution to this problem can be achieved through *bootstrapping*, as described below:

1. A basic, idealized stochastic FSM, F, which accounts for canonical natural sentences that would cover the semantics of the task involved in the application, is manually built by a human expert.
2. The system is started with F and a non-stochastic error model using Levenshtein distance[4] (see Eq. (1)). In this case, the probabilities of the FSM are ignored.
3. A given small initial set of natural sentences is used to bootstrap the system. For each natural sentence, if the simplification made by the EC parsing is considered adequate *by the human expert* then: i) the output "simplified" sentence is considered as the "*clean*" *input* of the noisy channel or the *simplified input*; ii) the user input sentence is considered as the "*distorted*" *output* of the noisy channel; and iii) the pair is added to a training corpus $\mathcal{T}$. Therefore, $\mathcal{T}$ is composed by a set of pairs (*user's natural sentence, adequate simplification*).
4. EM re-estimation of E operations is performed using $\mathcal{T}$.
5. Steps 3 and 4 are repeated, now performing *stochastic* (rather than Levenshtein) EC parsing based on the parameters estimated in step 4, until no more pairs are appended to $\mathcal{T}$.
6. The whole process is repeated with increasing amounts of new training natural language sentences, appending new training pairs to $\mathcal{T}$, until the performance of the system is considered satisfactory.

The overall convergence of this process critically depends on the rate of sentences at which an "adequate simplification" is obtained in step 3. This rate can be easily increased by using a K-best EC parser (Amengual, 1999; Amengual, 2000). In this way, the K-best solutions provided by the system to each input natural sentence are offered to the *human expert* who selects the ones that are considered adequate simplifications. Then, the pairs formed by the input sentence and all of its adequate simplifications (K, at most, from one to three in practice) can be appended to $\mathcal{T}$. It is likely that, during the bootstrapping process, some input sentences cannot be correctly simplified (even using the K-best solutions provided by the parser). This can be due to any of the following:

- They are out of the semantic domain defined by the task. Sentences of this kind are simply ignored.
- They contain contrived syntactic variations which cannot be adequately modeled with insertions, deletions and substitutions of words. In this case, nothing can be done. More powerful error model operations need to be studied.
- No adequate simplification is covered by the language accepted by the basic FSM.

In the latter case, these "residual" errors can now be used to "refine" F, yielding a new FSM, F_1. Obviously, the human expert could *manually* add the new transitions and states but this is costly. Also, maintaining the consistence of successive manual grammar modifications can be quite problematic. Another possibility is to ask the expert just to provide an adequate simplification for each of these sentences (manually, maybe employing some hints from the output of the EC parser) and to use the grammatical inference technique described in the following section. Thus, in the successive iterations of the above described bootstrapping procedure, if a few more residual errors are produced, new FSMs (F_2, F_3, ...) can be built upon demand.

Consider, for instance, the canonical FSM depicted in figure 2(a) (this figure indeed shows the working of the refinement algorithm explained in the next section). It allows for querying about the *names* and/or *lengths* of rivers that run through a given region, generically named REGION, following a specific (supposedly standard) syntax. Assume that a user inputs something like: "*Tell me what are the flows of the rivers that run through Valencia*". This sentence could not be suitably simplified since the basic language accepted by the FSM does not allow asking about the *flows* of rivers. When prompted to provide an appropriate simplification to be used as learning input to the refinement algorithm, the human expert should supply this "simplification": "*What are the flows of the rivers that traverse REGION*". This sentence would be the *actual* training input for the refinement algorithm. Keep in mind that the sentences used to refine the initial FSM have, in practice, (simple) syntactic structures whose meaning is not "covered" at all by any syntactic structure modeled by $L(F)$. Most natural language effects and variations should be accounted for by the error model.

As this procedure advances, the number of residual errors (for the "simplification" of which expert assistance is required) is expected to rapidly decrease as the quality of the models increases, allowing for automatic simplifications of more and more input sentences.

4. Grammatical inference through error-correcting

Error-correcting techniques have already been used in the field of Grammatical Inference (GI). This is the case of the *Error-Correcting Grammatical Inference* (ECGI) and other related algorithms (Chirathamjaree & Ackroyd, 1980; Rulot & Vidal, 1987, 1988; Thomason, Granum & Blake, 1986).

As was originally proposed, ECGI can only work with and produce *acyclic* FSMs. This was mainly due to the computational problems posed by the parsing of deletion transitions when using cyclic FSMs. A detailed description of problems of this kind can be found in

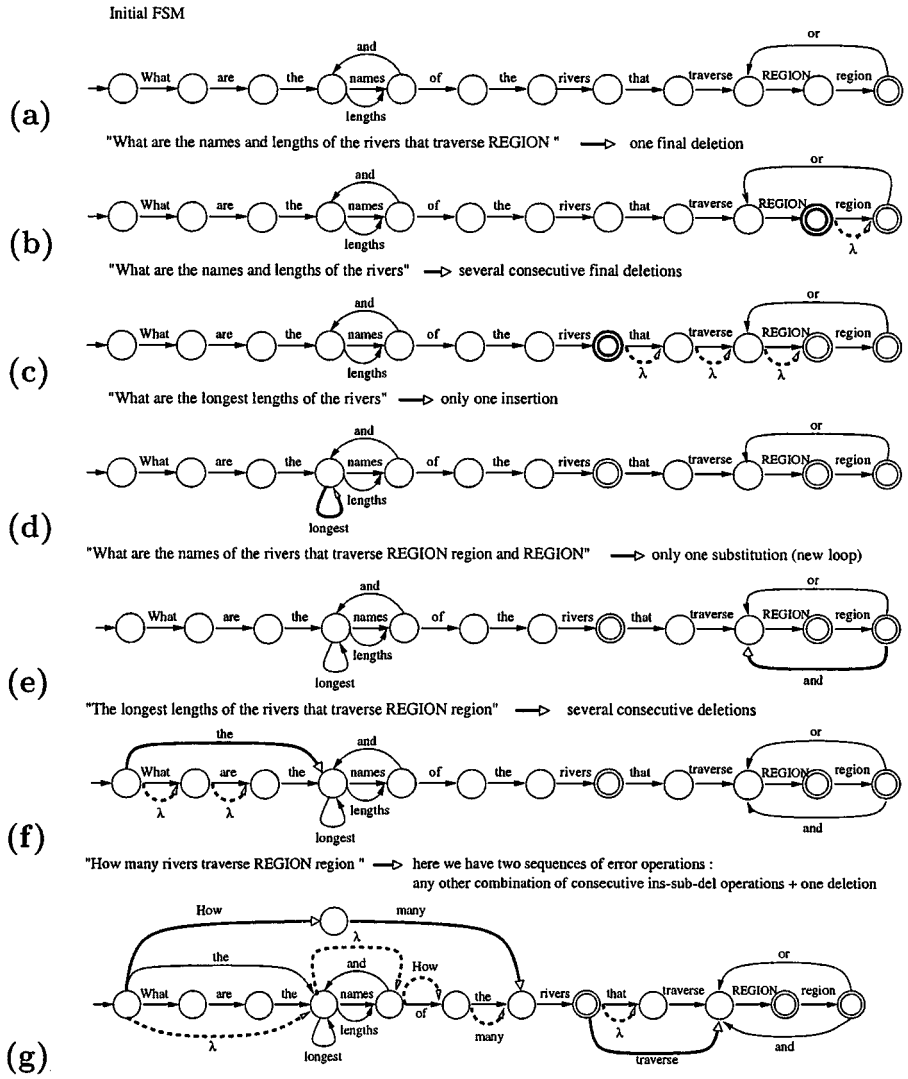

Figure 2. Example of FSM refinement using the ECGRA with a given sequence of six "simplified" learning strings.

Amengual (1999), Amengual and Vidal (1998), Gregor and Harris (1995), where parsing algorithms which efficiently solve these problems are proposed. These techniques allow for performing full error-correcting parsing using *any* kind of FSMs, which are no longer restricted to being acyclic (Amengual, 1999; Amengual & Vidal, 1998). These techniques constitute the core of a *new* ECGI-like inference algorithm, which is proposed here.

The algorithm starts with a given grammar (or equivalent FSM),[5] F, and uses a set of training strings to update this grammar according to the possibly new syntactic constructions

found in these strings. This is particularly useful in the application considered in this paper, where small refinements of a *given* grammar are required. The algorithm, which will be referred to as *Error-Correcting Grammar Refinement Algorithm* (ECGRA) works by performing the following actions with every new training string $\bar{z}$:

1. *EC Parsing* using the grammar to be refined. Since we are interested in adding the minimum number of states/rules required to accept $\bar{z}$, the minimum-distance EC framework is employed (therefore solving Eq. (1)). The dissimilarity function which best achieves this goal is Levenshtein distance. As a result of this process, the sequence of transitions that allows for generating the sentence which is closest to $\bar{z}$ is obtained. If $\bar{z} \notin L(F)$, this sequence contains error operations. Information is collected about:
   - *type* of error; in the case of substitutions and insertions, the input symbol involved in the corresponding error operation, $\bar{z}_i$, is also known.
   - *location*(*s*) in F where errors occurred; that is, one (or more) *pair*(*s*) *of states*, (q_1, q_2), where one (or more) sequence(s) of error operations was (were) produced during the parsing.
2. *FSM Updating* with new states and/or transitions associated to the sequence(s) of error operations found. The updating procedure for a given sequence of error operations, O, is shown in algorithm 1 (see also figure 2). Note that in our notation q_1 is always the first state and q_2 is always the final state in the sequence of error operations.

Algorithm 1 FSM updating procedure for a sequence of error operations

switch *error_type*(O)
  case one or several consecutive *final* deletions (i.e. $q_2 \in Z$):
    $Z = Z \cup \{q_1\}$
  case one or several consecutive deletions:
  $\forall a, q : (q_2, a, q) \in E$
    $E = E \cup \{q_1, a, q\}$
  end $\forall$
  case only one insertion:
    $E = E \cup \{q_1, \bar{z}_i, q_1\}$
  case only one substitution:
    $E = E \cup \{q_1, \bar{z}_i, q_2\}$
  case any other combination of consecutive ins-sub-del operations:
    A new path connecting q_1 with q_2 *is created in the FSM.*
    This path only contains the transitions which are associated
    to the insertion and substitution errors occurred.
end_switch

Note that this algorithm only refines the structure of F. If the initial FSM is stochastic, then probabilities are simply ignored. After the ECGRA refinements, maximum likelihood re-estimation techniques (Casacuberta, 1996) can be applied on the modified automaton. Another possibility, which is adopted in this work, consists in simply performing a *flat*

smoothing: a given small quantity is discounted from the consolidated transitions, which is then distributed among the added ones.

An example of the working of ECGRA is shown in figure 2. In this figure, consolidated transitions and states of the successively refined grammars are drawn as thin lines while the error operations used during the EC parsing at each step are shown with *dashed* bold lines. The states and transitions *actually* added by the ECGRA are represented by bold lines.

One drawback of this algorithm with regard to the original ECGI lies in the fact that an *initial* grammar is needed as input, since the heuristics employed by the ECGRA have been specifically designed to refine a *given* grammar (otherwise, they easily tend towards over-generalization, see algorithm 1). Nevertheless, future developments need to explore the possibility of working only on a single training corpus.

5. Description of a language simplification task

The task considered here is a standard task for testing Language (speech) Understanding systems called MGEO[6] which is defined in the framework of the ALBAYZIN Spanish language benchmarking project (Díaz-Verdejo et al., 1998). Taking all of the available information about the general task domain into account (Díaz-Verdejo et al., 1998), the reasons for the selection of the MGEO task are twofold. On the one hand, MGEO is a semantically constrained task and, on the other hand, no syntactic restrictions are imposed to the construction of the queries. In this case, the kind of information that the database can be asked defines the semantics of the task. The database is assumed to contain information about (this is the semantic scheme of the database):

- *entities*: seas, regions and rivers.
- *attributes* of these entities: names of the seas, names and areas of the regions and names, lengths and flows of the rivers.
- *relations* among these entities: flow_into(river, sea), end(river, region), source(river, region), traverse(river, region), surround(sea, region).

Correspondingly, queries to the MGEO database are expected to include questions about:

1. entities or attributes of an entity.
2. entities related to another entity.
3. attributes of an entity (or entities) related to another entity.
4. any combination of the previous ones.

5.1. Purpose of language simplification in the MGEO task

Although the number of possible types of queries to the database is small from a semantic point of view, the syntactic variability that can be exhibited by real user queries is enormous. In other words, there exist a *many-to-one* mapping between the syntactic and semantic domains of the MGEO task. However, most of the syntactic differences with regard to what could be considered as *standard* syntactic constructions to make queries to the database

consist in (more or less small) variations in vocabulary, the use of superfluous words, interjections, etc. mainly due to user oral/writing habits.

Clearly, it is not realistic to try to model (with a regular or even context-free grammar) all these syntactic peculiarities/habits exhibited by casual users. Therefore, in practice, the performance of even the best trained models drops dramatically when tested with spontaneous natural language inputs. For instance, suppose that our language model has the syntactic structures "*What seas surround Spain*" and "*What seas surround all the regions*" for referring to the database query (semantic concept) *List seas surround Spain*. However, if a user inputs "*Tell me the seas that surround the Iberian Peninsula*"[7] the system fails and no answer is returned to the user.

Under the Language Simplification framework, the most likely sentence of our language model that corresponds with the user input is "*What seas surround Spain*", therefore providing the user with the right answer to his/her query. The error model considers that "*What*" has been substituted by "*Tell*" and "*Spain*" by "*Peninsula*", that "*seas*" and "*surround*" have been substituted for themselves, and, finally, that "*me*", "*the*", "*that*", "*the*", and "*Iberian*" have been inserted.

5.2. Baseline language model

In order to obtain a first canonical language model of the MGEO task, an initial stochastic FSM was (manually) built by an expert. This model attempted to cope with only basic variability in the construction of the queries by means of canonical syntactic forms. These canonical forms were designed to convey the whole semantics of the task. In addition, both the specific proper nouns (seas, regions and rivers) and the numbers (area of the regions, length and flow of the rivers, . . .) were respectively labelled in the FSM using four categories: SEA, REGION, RIVER and NUM. Examples of canonical syntactic forms (that is, sentences accepted by the basic FSM) corresponding to each of the above mentioned four types of expected queries are:

1. *cuál es la longitud y el caudal del río RIO*
   (what is the length and flow of the RIVER river)
2. *cuántos ríos desembocan en el mar MAR*
   (how many rivers flow into the SEA sea)
3. *cuál es la extensión de las comunidades que atraviesa el río que tiene máxima longitud*
   (what are the areas of the regions that are traversed by the river that has maximum length)
4. *cuál es la longitud del río que tiene mínimo caudal y atraviesa más comunidades y desemboca en el mar que baña COM*
   (what is the length of the river that has least flow and traverses the greatest number of regions and flows into the sea that surrounds REGION)

This FSM, which had 230 states, 665 transitions, and a lexicon consisting of 60 words, was the starting point for the GI refinement ECGRA described in Section 4. Note that the task of this algorithm is simply to refine the syntax and possibly some of the lexicon of the basic *canonical* model. Important vocabulary variations regarding this model, including a large number of words used to instantiate the four lexical categories SEA, REGION, RIVER

and NUM, are expected to be exhibited by natural language sentences. These variations, along with many non-standard syntax inflections, irregularities, etc., are expected to be (stochastically) modeled by the error model, E.

6. Experiments and results

The whole corpus of the MGEO task (see Section 5) consists of 908 sentences (806 different sentences) with a lexicon of 307 words from a total of 9,008 running words. Two *independent* sets were extracted from this corpus; one for training with 700 *randomly chosen* sentences and one for test with the remaining 208 sentences.

In order to *simulate* the bootstrapping procedure described in Section 3, the training set was partitioned into *seven* blocks, each having 100 sentences: sentences 1–100 in the first block, sentences 101–200 in the second block, ..., sentences 601–700 in the seventh block. Our Language Simplification system for the MGEO task was started with the baseline language model described in Section 5.2 and employed the Levenshtein distance to give values to E operations. The following incremental learning process was thus carried out, starting with the first block and ending with the seventh block:

1. The *bootstrapping* procedure explained in Section 3 was performed in order to estimate the stochastic values of E operations that allow for achieving right simplifications. A K-best EC parser with $K = 10$ was used. The human expert selected among the solutions provided by the parser those considered as adequate simplifications in order to build the required training corpus, $\mathcal{T}$, composed by "distorted"-"clean" pairs. As commented on in Section 3, this process is *repeated* until no more training pairs are appended to $\mathcal{T}$.
2. Those sentences that could not be correctly simplified during the incremental estimation of E operations (residual errors) were collected. Then, their corresponding simplified versions were manually supplied by the human expert (often following the hints provided by the output of the EC parser). These simplifications were finally employed to refine the language model used in the previous step (the baseline language model when using the first block of training sentences) by means of the ECGR algorithm explained in Section 4.
3. The whole process is again performed with the next training block, using the E operation estimates achieved in step 1 and the refined FSM obtained as a result of step 2 (flat smoothing was performed to give probabilities to the added transitions).

Table 1 shows the number of iterations carried out in step 1 of the above described incremental learning process and the number of sentences that were not correctly simplified, along with the sizes of the successively refined FSMs with each of the seven training blocks.

The charts displayed in figure 3 show the improvement achieved with progressive E operation estimation (step 1 of the incremental learning process) in the first, fourth, and seventh training blocks. When comparing first to last iteration for each of these blocks, it can be observed that not only did the total number of sentences which were correctly simplified increase, but their position in the K-best ranking also rose. Keep in mind that the baseline language model and the Levenshtein distance were employed in the *first* iteration with the *first* training block.

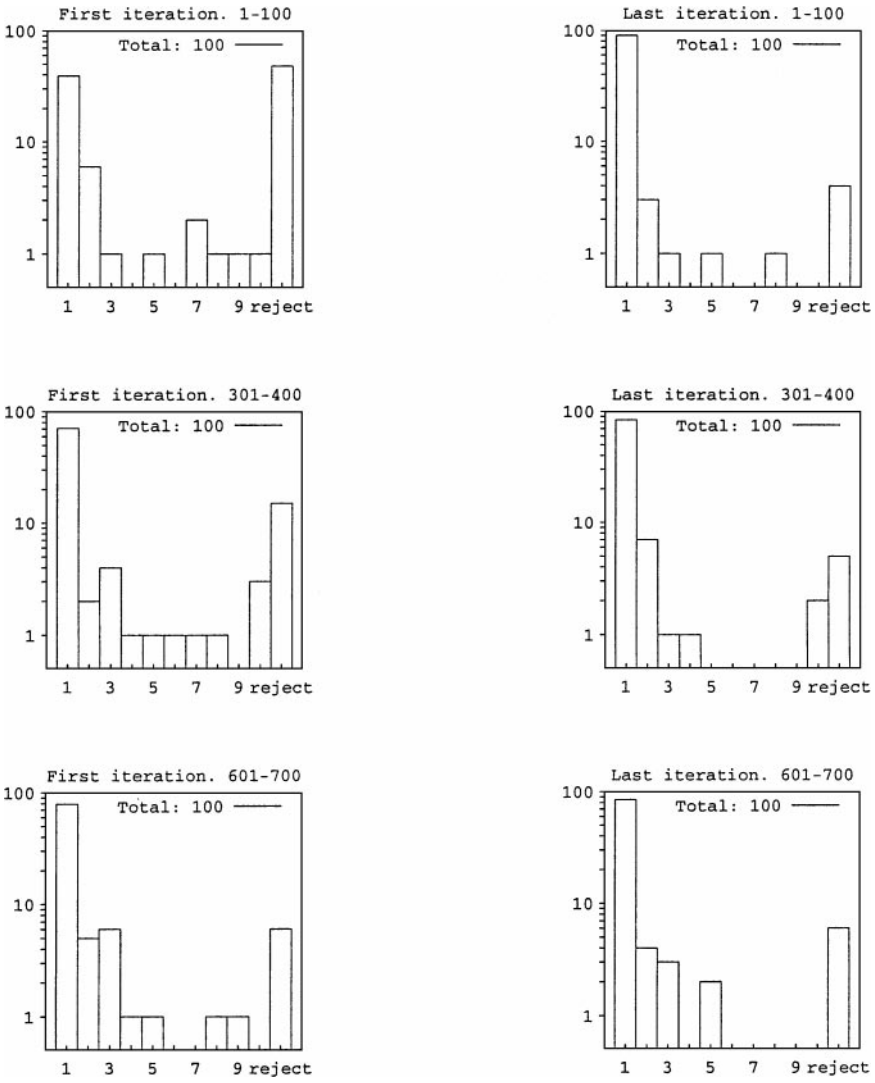

Figure 3. Number of times the adequate simplification was obtained in each of the 10 solutions provided by the K-best EC parser. First and last iterations (see also Table 1) with the *first*, *fourth*, and *seventh* training blocks are shown in this figure.

To finish the bootstrapping training phase, step 1 of the incremental learning process was again performed with the 42 sentences to which a correct simplification was not provided (see Table 1). The estimates computed for E during the previous training with the seven blocks and the refined FSM finally obtained were used. As a result, this time 12 of these sentences were adequately simplified and the *final* probabilities for E operations were yielded. Therefore, 670 training sentences (95.71%) were correctly simplified and 30

Table 1. Number of E operation estimation iterations and of residual errors (sentences not adequately simplified) in step 1, along with the sizes of the FSMs refined in step 2 with each of the 7 training blocks. The initial FSM had 230 states, 665 transitions, and 60 words.

Training Block	1–100	101–200	201–300	301–400	401–500	501–600	601–700
Iterations	6	6	3	3	3	2	2
Residual errors	4	5	8	5	10	4	6
States	243	253	265	283	285	286	287
Transitions	692	710	746	785	789	793	795
Lexicon (words)	61	63	63	65	65	65	65

Table 2. Some examples of simplifications.

Sentence	dime la longitud y caudal del río Ebro (tell me the length and flow of the Ebro river)
Simplification	cuál es la longitud y el caudal del río RIO (what is the length and flow of RIVER river)
Sentence	¿cuál es el número de ríos que desembocan en el mar Mediterráneo? (what is the number of rivers that flow into the Mediterranean sea?)
Simplification	cuántos ríos desembocan en el mar MAR (how many rivers flow into the SEA sea)
Sentence	¿cuál es el río de mayor caudal que desemboca en el mar Mediterráneo? (what is the river with the greatest flow that flows into the Mediterranean sea?)
Simplification	cuál es el nombre del río que tiene máximo caudal y desemboca en el mar MAR (what is the name of the river that has the greatest flow and flows into the SEA sea)

(4.29%) were "rejected". Examples of sentences with their adequate simplifications output by the system are shown in Table 2. As a final step, the probabilities of the final refined FSM were estimated through a Viterbi re-estimation procedure (Casacuberta, 1996) using all of the correct *simplifications* obtained during the whole training process (therefore ignoring the probabilities previously used).

Once the training (refinement in the case of the FSM, estimation in the case of error model operations) of F and E was completed, the performance of the system was checked by the human expert using the independent previously unconsidered 208 *test* sentences. Table 3 summarizes the results achieved in the test as assessed by the expert. This table shows the percentage of correctly simplified sentences for increasing values of K ($K = 10$) in the ranking of the K-best solutions provided by the EC parser.

Table 3. Rate of correctly simplified test sentences (208) taking each of the K-best solutions provided by our LS system into account.

Value of K	1	2	3	4	5	6	7	8	9	10
Success rate (%)	79.8	82.2	84.1	85.0	86.5	87.0	88.9	89.4	89.4	89.9

7. Final remarks

Grammatical Inference (GI) along with Error-Correcting (EC) techniques have been introduced in this work to deal with a novel view of Language Understanding (LU). Under this view, the meaning of an input sentence is assumed to be associated with a *canonical* natural-language-like sentence belonging to a simple grammar that models elementary syntax. This basic syntax allows that the meaning can be expressed in the considered task. Such elementary representations of the meaning readily allow for simple implementations of the final drivers that could ultimately perform the actions requested by the user input sentences.

Bootstrapping methods have been used to incrementally refine the coverage of a basic initial grammar while progressively estimating the probabilities for the operations of the corresponding error model. Effective training techniques have been described to estimate EC operations and a new GI technique based on Error Correction has been introduced to solve the grammar refinement problem.

Empirical results on a moderately complex task show that the proposed approach is promising as a new paradigm for a number of Language Processing applications, including LU. Future work in this direction should encompass developing more complex error models together with their corresponding probability estimation procedures. Also, improved grammar refinement algorithms should result in faster bootstrapping training, thus reducing the need for human expert assistance. Finally, further simplification of human expert intervention can be expected by extending the proposed techniques to be able to work with *context-free* rather than finite state models.

Acknowledgments

The authors wish to thank the anonymous reviewers for their careful reading of the manuscript and for their useful suggestions and valuable comments. Work partially funded by the European Union and the Spanish CICYT under contracts IT-LTR-OS-30268 and TIC98-0423-CO6/05.

Notes

1. SEA stands for any body of water in the database.
2. The operation $(\backslash, \backslash)$ is not allowed in E.
3. In the sense of preserving the meaning of the original user input.
4. That is, a minimum-distance EC parser.
5. The concepts grammar/FSM, generate/accept, transition/rule and state/non-terminal will be equally used throughout this section.
6. Spanish acronym of *Access to a Geographical Miniature Database.*
7. This is a real example.

References

Amengual, J. C. (1999). *Técnicas de Corrección de Errores y su Aplicación en Reconocimiento de Formas, Tratamiento del Lenguaje Natural y Traducción Automática.* Ph.D. thesis, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia (Spain), in Spanish.

Amengual, J. C. (2000). An a^* search-based k-best error-correcting viterbi parser. Technical Report DI 02-03/00, Unidad Predepartamental de Informática, Universidad Jaume I, Castellón (Spain).

Amengual, J. C., & Vidal, E. (1998). Efficient error-correcting viterbi parsing. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 20:10*, 1109–1116.

Amengual, J. C., & Vidal, E. (2000). On the estimation of error-correcting parameters. Technical Report DI 01-03/00, Unidad Predepartamental de Informática, Universidad Jaume I, Castellón (Spain).

Baum, L. E., & Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin American Mathematical Society, 73*, 360–363.

Bunke, H., & Csirik, J. (1995). Parametric string edit distance and its application to pattern recognition. *IEEE Trans. on Systems, Man, and Cybernetics, 25:1*, 202–206.

Casacuberta, F. (1996). Growth transformations for probabilistic functions of stochastic grammars. *International Journal of Pattern Recognition and Artificial Intelligence, 10:3*, 183–201.

Chirathamjaree, C., & Ackroyd, M. H. (1980). A method for the inference of non-recursive context-free grammars. *Int. Journal Man-Machine Studies, 12*, 379–387.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Royal Statistical Society, 39:1*, 1–38.

Díaz-Verdejo, J. E., Peinado, A. M., Rubio, A. J., Segarra, E., Prieto, N., & Casacuberta, F. (1998). Albayzin: A task-oriented spanish speech corpus. In *First International Conference on Language Resources and Evaluation* (pp. 497–501), Granada (Spain).

Fu. K. S. (1982). *Syntactic Pattern Recognition and Applications*. Englewood Cliffs, New Jersey: Prentice Hall.

Gonzalez, R. C., & Thomason, M. G. (1978). *Syntactic Pattern Recognition. An Introduction*. Reading, Massachusetts: Addison-Wesley.

Gregor, J., & Harris, R. S. (1995). String matching with left-to-right networks. *Pattern Recognition Letters, 16*, 213–218.

Kruskal, J. B. (1983). An overview of sequence comparison. In D. Sankoff & J. B., Kruskal (Eds.), *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison* (pp. 1–44). Reading, Massachusetts: Addison-Wesley.

Rice, S. V., Bunke, H., & Nartker, T. A. (1997). Classes of cost functions for string edit distance. *Algorithmica, 18*, 271–280.

Ristad, E. S., & Yianilos, P. N. (1998). Learning string-edit distance. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 20:5*, 522–532.

Rulot, H., & Vidal, E. (1987). Modelling (sub)string-length-based constraints throught a grammatical inference method. In P. A. Devijver & J. Kittler (Eds.), *Patter Recognition: Theory and Applications* (pp. 451–459), Springer-Verlag.

Rulot, H., & Vidal, E. (1988). An efficient algorithm for the inference of circuit-free automata. In G. Ferraté, T. Pavlidis, A. Sarfeliu, & H. Bunke (Eds.), *Syntactic and Structural Pattern Recognition* (173–184), Springer-Verlag.

Sankoff, D., & Kruskal, J. B. (Eds.) (1983). *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*. Reading, Massachusetts: Addison-Wesley.

Thomason, M. G., Granum, E., & Blake, R. E. (1986). Experiments in dynamic programming inference of markov networks with strings representing speech data. *Pattern Recognition, 19:5*, 343–351.

Vidal, E., Casacuberta, F., & García, P. (1995). Grammatical inference and automatic speech recognition. In A. J. Rubio & J. M. López (Eds.), *Speech Recognition and Coding, New Advances and Trends* (pp. 174–191), NATO Advanced Study Institute. Berlin: Springer-Verlag.

Received December 9, 1998
Revised March 22, 2000
Accepted May 23, 2000
Final manuscript May 31, 2000