KYBERNETIKA — VOLUME 28 (1992), NUMBER 1, PAGES 69 – 76

SEVERAL COMMENTS ON PATTERN RECOGNITION SYSTEM BASED ON THE USE OF ATTRIBUTED GRAMMARS

JIŘÍ KEPKA

In the paper attributed grammars as a tool for combining syntactic and statistical approaches to pattern recognition is discussed. The points where the algorithm for pattern classification based on the use of attributed grammars can be efficiently implemented in parallel are outlined. Some advantages of the use of declarative programming languages like Prolog for the realization of pattern recognition system based on attributed grammars are shown.

1. INTRODUCTION

It is known that the syntactic approach to pattern recognition provides a capability for describing the details of internal structure of a pattern. But the sensibility to noise which usually causes the structural change in a pattern makes this approach alone inadequate for some practical applications. On the other hand it is also known that the statistical approach to pattern recognition is unable to describe pattern structures and subpattern relations. The fact that the advantage of one approach is at the same time the disadvantage of the other makes the idea of a hybrid model which would incorporate the advantages of both very attractive. There have been proposed some techniques how to combine both approaches, e. g. applications of stochastic and/or transform grammars, stochastic error-correcting syntax analysis [2], attributed grammars [1]. Especially, attributed grammars are successfully used as the model for pattern grammar because of their descriptive power, which is due to their ability to handle syntactic as well as semantic information, see e. g. [3], [4].

In this paper the use of stochastic attributed grammars as a tool for combining syntactic and statistical approaches to pattern recognition is discussed. Several possibilities of parallel implementation of the system based on attributed grammars are shown.

At first, basic notations and definitions of stochastic attributed grammars are briefly reviewed.

2. BASIC NOTATIONS AND DEFINITIONS

Definition. A stochastic attributed context-free grammar is given by a 4-tuple $G = (V_N, V_T, P, S)$ where V_N is the set of nonterminals, V_T is the set of terminals, $S \in V_N$ is the starting symbol, for each $X \in (V_N \cup V_T)$, there exists a finite set of attributes $A(X)$, each attribute α of $A(X)$ having a set, either finite or infinite, of possible values D_α; and P is a set of productions each of which is divided into two parts: a syntactic rule and a semantic rule. The syntactic rule is of the following form

$$X_0 \xrightarrow{p} X_1 X_2 ... X_m$$

where $X_0 \in V_N$ and each $X_i \in V_N \cup V_T$ for $1 \leq i \leq m$, p is the probability associated with this syntactic rule, $0 < p \leq 1$. The summation of all the probabilities associated the syntactic rules with X_0 at the left-hand side must be equal to one. The semantic rule is a set of expressions of the following form

$$\begin{aligned} \alpha_1 &\rightarrow f_1(\alpha_{11}, \alpha_{12}, ..., \alpha_{1n_1}) \\ \alpha_2 &\rightarrow f_2(\alpha_{21}, \alpha_{22}, ..., \alpha_{2n_2}) \\ &\vdots \\ \alpha_n &\rightarrow f_n(\alpha_{n1}, \alpha_{n2}, ..., \alpha_{nn_n}) \end{aligned}$$

where $\{\alpha_1, \alpha_2, ..., \alpha_n\} = A(X_0) \cup A(X_1) \cup ... \cup A(X_m)$, each α_{ij} $(1 \leq i \leq n,\ 1 \leq j \leq n_i)$ is an attribute of some X_k for $0 \leq k \leq m$, and each f_i $(1 \leq i \leq n)$ is an operator which may be in one of the following three forms:

1. a mapping f_i: $D_{\alpha_{i1}} \times D_{\alpha_{i2}} \times ... \times D_{\alpha_{in_i}} \rightarrow D_{\alpha_i}$
2. a closed form function
3. an algorithm which takes $\alpha_{i1}, \alpha_{i2}, ..., \alpha_{in_i}$ and any other available information or data as input and α_i as output [1].

$A(X)$ is partioned into two disjoint sets, the synthesized attribute set $A_0(X)$ and the inherited attribute set $A_1(X)$; $\alpha \in A(X)$ has a set of possible values D_α, from which one value will be selected for each appearance of X in a derivation tree.

An input pattern is first preprocessed and all necessary primitives and their attributes are extracted. Then a structural representation is formed by assigning symbols to primitives, selecting e. g. concatenating directions, and any other prespecified relations. The string, i. e. the resulting representation; is then parsed by using the syntactic rules of the attributed grammar, while the semantics computations are performed at the same time. It should be mentioned here that to obtain all required nonterminal (subpattern) attributes (according to the semantic rules) it is sometimes necessary to go back to the input pattern because some subpattern attributes can not be obtained by a computation from lower level terminal attributes and thus the subpattern corresponding to the

nonterminal must be found out and the corresponding subpattern attributes must be extracted, see Figure 1.

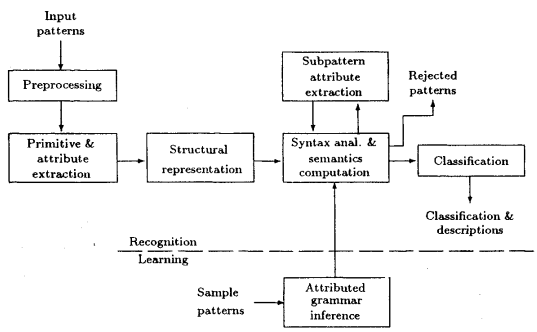

Figure 1: Pattern analysis system using attributed grammars.

As the authors in [1] point out the subpattern attributes cannot be extracted before syntax analysis and semantics computations because without the guidance of syntax analysis, the system would not know which terminals (primitives) should be grouped into a subpattern (nonterminal). This is an advantage of using attribute grammars because subpattern attribute extraction now becomes more effective with the guidance of syntax analysis. Such an advantage is not obtainable by using the statistical approach alone. The results of syntax analysis are both a parse of the analyzed pattern and its total attribute vector if the analyzed pattern is syntactically correct. Then the pattern is classified in accordance with its total attribute vectors.

Remark. The introduction of attributes into a grammar usually makes the reduction of grammatical complexity possible, i.e. an attributed grammar can describe the structure of patterns of several classes because the classification is performed on the total attribute vector (for details see [1]). In the case that more (than one) attributed grammars are necessary to describe all kinds of patterns syntax analysis can be performed simultaneously according to all grammars because it lends itself readily to parallel implementation.

3. THE USE OF ATTRIBUTED GRAMMARS FOR STATISTICAL CLASSIFICATION

As it is well known from the literature (cf. e. g. [1]), at least three kinds of statistical information should be used for classification.

1. The occurrence probability of every pattern class.
2. The occurrence probability distribution of the total attribute vector of each pattern.
3. The occurrence probability of a specific pattern structure within its pattern class.

While the occurrence probability of every pattern class usually is determined intuitively or through longer observations of pattern occurrences, the occurrence probability of a specific pattern structure within its pattern class is computed as the product of all the probabilities associated with the grammatical rules used in parsing the string (the pattern representation). It should be reminded here that attributed grammar can usually describe several pattern classes as it was noted in the above remark and hence the probabilities associated with the syntactic rules are generally not equal for different classes. If the parallel computation can be performed, these probabilities $P(z|C_1), P(z|C_2), ..., P(z|C_N)$ where N is the total number of classes and z is the string representation of the given input pattern can be computed simultaneously.

After an input pattern ω is successfully parsed a total attribute vector X is also obtained as the result of simultaneous semantic computation. In [1] it is noted that if a top-down parsing is adopted to analyze pattern structures, the inherited attributes are more convenient for use because they can be computed in a top-down fashion, starting from the start symbol S of the grammar. On the contrary, if a bottom-up parsing is preferred, the synthesized attributes should be used, which are computed in a bottom-up fashion. The trouble with the computation when both the synthesized and the inherited attributes are necessary can be sometimes efficiently solved if a declarative programming language like Prolog is used. Many predicates (not all!) can be used to perform several functions depending on how the predicate is called. In one situation, a particular parameter may have a known value; in a different situation some other parameter may be known; and for certain purposes all of the parameters may be known at the time of the call. When a Prolog clause is able to handle multiple flow patterns (i. e. statuses of the arguments to a given predicate), it is known as an invertible clause; e. g. the *append* predicate; for more details see [5].

3.1. An optimum decision rule for pattern classification using attributed grammars

First given an unambiguous attributed grammar G, i. e. each string generated by G has only one derivation, let N be the total number of pattern classes covered by G, and n be the total number of distinct strings (corresponding to pattern structures) which can be generated by G. It is considered that each pattern class C_i, $i = 1, 2, ..., N$,

syntactically contains all the n strings $z_1, z_2, ..., z_n$ generated by G. Let us consider the 2-tuple $\omega_{ij} = (z_j, X_{ij})$ as the description of a noise-free pattern in class C_i, each class C_i consists of exactly n noise-free patterns $C_i = \{\omega_{i1}, \omega_{i2}, ..., \omega_{in}\}$ where subscripts i, j in X_{ij} specify that this X_{ij} is computed for z_j of class C_i. Each ω_{ij} semantically can be deformed into a finite or infinite set of noisy versions because of noise and distortion. All these versions have an identical symbolic string representation z_j but attributes of input patterns are subject to numerical variations due to noise and distortion. Thus for $\omega_{ij} = (z_j, X_{ij})$ in C_i let the set of its noisy versions be denoted $D(\omega_{ij}) = \{\omega_{ijk} \mid \omega_{ijk} = (z_j, X_{ijk}), k = 1, 2, ..., m_{ij}\}$ where m_{ij} may be finite or infinite, then C_i can be regarded as $C_i = D(\omega_{i1}) \cup D(\omega_{i2}) \cup ... \cup D(\omega_{in})$. Since each $\omega_{ijk} \in D(\omega_{ij})$ may occur with a different probability, we can introduce a conditional probability distribution on $D(\omega_{ij})$ for class C_i such that

$$p(\omega_{ijk}|\omega_{ij}, C_i) = p(X_{ijk}|z_j, C_i)$$

is the occurrence probability for $\omega_{ijk} = (z_j, X_{ijk}) \in D(\omega_{ij})$, which will be called the attribute occurrence probability of ω_{ijk} from ω_{ij}, in contrast with the structural occurrence probability of z_j within class C_i, $P(z_j|C_i)$ [1].

If the string representation of an unknown pattern ω is accepted by the grammar as z_j, and if the computed total attribute vector (TAV) for ω is X, the tuple $\omega = (z_j, X)$ can be regarded as a noisy version of ω_{ij}, included in $D(\omega_{ij})$. Then the attribute occurrence probability of ω from ω_{ij} with respect to class C_i is

$$p(\omega|\omega_{ij}, C_i) = p(X|z_j, C_i)$$

and hence the composite occurrence probability that $\omega \in C_i$ is

$$\begin{aligned} p(\omega|C_i) &= p(\omega|\omega_{ij}, C_i)P(\omega_{ij}|C_i) \\ &= p(X|z_j, C_i)P(z_j|C_i). \end{aligned}$$

With respect to the assumption that the attributed grammar is unambiguous, it is possible to compute the a posteriori probability $p(C_i|\omega)$ that ω can be recognized as a deformed pattern from the pattern class $C_i = \bigcup_{j=1}^{n} D(\omega_{ij})$

$$\begin{aligned} p(C_i|\omega) &= p(\omega|C_i)P(C_i)/p(\omega) \\ &= p(\omega|\omega_{ij}, C_i)P(\omega_{ij}|C_i)P(C_i)/p(\omega) \\ &= p(X|z_j, C_i)P(z_j|C_i)P(C_i)/p(\omega) \end{aligned}$$

where $\omega_{ij} = (z_j, X_{ij})$, $P(C_i)$ is the a priori probability of class C_i, and $p(\omega)$ can be calculated as $p(\omega) = \sum_{i=1}^{N} p(\omega|C_i)P(C_i)$.

When the attributed grammar is ambiguous, the above discussion is no longer valid as the authors in [1] point out and it should be modified. For the system to be implementable, G is assumed not to be infinitely ambiguous that means there is a finite number of distinct derivations for each string accepted by G. This assumption is general

enough for most applications. In this case the following equations hold [1]:

$$p(\omega|C_i) = \sum_{k=1}^{l_j} [p(\omega|\omega_{ij}^k, C_i)\ P(\omega_{ij}^k|C_i)]$$
$$= \sum_{k=1}^{l_j} [p(X^k|z_j^k, C_i)\ P(z_j^k|C_i)] \quad (1)$$

After applying Bayes Theorem on (1):

$$assign\ \omega\ to\ C_M \quad if \quad \sum_{k=1}^{l_j} [p(X^k|z_j^k, C_M)\ P(z_j^k|C_M]P(C_M)$$
$$= \max_{i=1,2,\ldots,N} \sum_{k=1}^{l_j} [p(X^k|z_j^k, C_i)\ P(z_j^k|C_i)]P(C_i)$$

where $p(\omega)$ common to both sides, has been dropped, l_j is the total number of distinct derivations. They can also be adjusted for parallel computation without serious difficulties. It should be mentioned here that because of the different combinations of syntactic rules used in the derivations, each different derivation z_j^k of $z = z_j$ will also result in a different structural occurrence probability value, which is denoted as $P(z_j^k|C_i)$. The above Bayes classification rule is for one grammar only. In the case of several grammar the extension is: assign ω to C_M if

$$p(C_M|\omega) = \max_{G_j} \max_{i=1,2,\ldots,N_{G_j}} p(C_i|\omega),$$

where N_{G_j} is the total number of pattern classes covered by grammar G_j. For more details in this subsection see [1]. It is evident that the last expression also lends itself readily to parallel implementation.

3.2. The difficulties with the computation of the attribute probability $p(X^k|z_j^k, C_i)$

The distribution $p(X^k|z_j^k, C_i)$ depends on the syntactic structure of the recognized string representation z_j. Given two input patterns from class C_i, assume that they have the same total attribute vector X^k computed. If they are accepted syntactically to have different string structures z_{j_1} and z_{j_2}, the probabilities or densities used for them should be $p(X^k|z_{j_1}^k, C_i)$ and $p(X^k|z_{j_2}^k, C_i)$, respectively, they may be different; but since in general the string structure (z_{j_1} or z_{j_2}) can be determined only after parsing, attribute occurrence probabilities can not be computed during the parsing procedure. Computation of such values must be delayed until parsing is completed and the structure of the input string representation is identified as the authors in [1] mention.

This problem exists whenever probability distributions or density functions for primitives (terminals) or subpatterns (nonterminals) are used for pattern classification. So

far, it has always been assummed that the occurrence probability of a certain primitive (terminal) is invariant with respect to different patterns. Such an assumption $p(X^k|z^k_{j_1}, C_i) = p(X^k|z^k_{j_2}, C_i) = p(X^k|C_i)$ though simplifying statistical discussion, theoretically is not general enough for applications using attributed grammars [1].

In the cited paper the authors added "identification rules" to identify the string structure. With the help of these identification rules correct probability density functions can be chosen after parsing is completed and then the attribute occurrence probability or density can be computed.

Another way how to solve this problem is the division of an attributed grammar into several ones so that each grammar will generate a set of strings $z_1, z_2, ..., z_n$ which satisfy the following conditions

$$p(X^k|z^k_1, C_i) = p(X^k|z^k_2, C_i) = ... = p(X^k|z^k_n, C_i) \tag{2}$$

But as the authors in [1] point out this solution is impractical because it destroys the essence of grammar usage – with one grammar efficiently covering as many structuraly similar patterns as possible.

Another way how to solve this problem can be suggested when parallel implementation is available. If the set of strings $\{z_i,\ i = 1, ..., n\}$ is finite and n is a small number the input string (pattern) z can usually be directly compared with possible strings $z_i,\ i = 1, ..., n$, and when the match arises the corresponding conditional probability or density functions are found. In the more complex cases (the number of generated strings by the grammar is large or infinite) it should be pointed out that there must exist only a finite number probability distributions or density functions for each terminal (primitive) for the problem to be relevant. In other words it must be possible to find a finite number of types of string structures within which (2) holds (n may be infinite because each type of string structure can have infinite different realizations). Then the amount of all possible results of computing $p(X^k|z^k_j, C_i)$ before the string structure identification at the end of parsing is also finite and in practice all possible results can usually be computed simultaneously during parsing in the case of parallel implementation because the introduction of attributes into a grammar reduces grammatical complexity and simplifies parsing. After parsing the string structure identification is performed (e. g. with the help of identifications rules) and corresponding result is chosen according to it. In this way the computation of attribute occurrence probabilities need not be delayed until parsing is completed and the pattern classification can be speeded up.

4. CONCLUSIONS

The attributed grammars are shown as an interesting tool for combining syntactic and statistical approaches to pattern recognition. Their advantages are most outstanding in the cases when pattern classes can be divided into groups, each group consisting of several pattern classes which are similar in structure but different in attributes.

When the declarative method instead of customary procedural one is used both inherited and synthesized attributes can often be used.

Several possibilities how to speed up pattern classification by parallel implementation are outlined.

(Received October 17, 1989.)

REFERENCES

[1] W. H. Tsai and K. S. Fu: Attributed grammar – a tool for combining syntactic and statistical approaches to pattern. IEEE Trans. Systems Man Cybernet. *10* (1980), 12, 873 – 885.

[2] S. Lu and K. S. Fu: Stochastic error-correcting syntax analysis for recognition of noisy patterns. IEEE Trans. Comput. *26* (1977), 12, 1268 – 1276.

[3] P. Trahanias and E. Skordalakis: Syntactic pattern recognition of the ECG. IEEE Trans. Patt. Anal. Mach. Intell. *12* (1990), 7, 648 – 657.

[4] E. Pietka: Feature extraction in computerized approach to the ECG analysis. Pattern Recognition *24* (1991), 2, 139 – 146.

[5] Turbo Prolog – the natural language of artificial intelligence. Owner's Handbook. Borland International 1986.

Ing. Jiří Kepka, Ústav teorie informace a automatizace ČSAV (Institute of Information Theory and Automation – Czechoslovak Academy of Sciences), Pod vodárenskou věží 4, 182 08 Praha 8. Czechoslovakia.