Syntactic Pattern Recognition

By
Nicolette Nicolosi
Ishwarryah S Ramanathan
Syntactic Pattern Recognition

- Statistical pattern recognition is straightforward, but may not be ideal for many realistic problems.
  - Patterns that include structural or relational information are difficult to quantify as feature vectors.
- Syntactic pattern recognition uses this structural information for classification and description.
- Grammars can be used to create a definition of the structure of each pattern class.
Classification

- Producing a classification can be done based on a measure of structural similarity in patterns.
- Each pattern class can be represented by a structural representation or description.
- It is often difficult to classify patterns that contain a large number of features.
Description

- A description of the pattern structure is useful for recognizing entities when a simple classification isn’t possible.
- Can also describe aspects that cause a pattern to not be assigned to a particular class.
- In complex cases, recognition can only be achieved through a description for each pattern rather than through classification.
When to Use It

- Picture recognition and scene analysis are problems in which there are a large number of features and the patterns are complex.
  - For example, recognizing areas such as highways, rivers, and bridges in satellite pictures.

- In this case, a complex pattern can be described in terms of a hierarchical composition of simpler subpatterns.
Hierarchical Approach

- The hierarchical approach comes from the similarity that can be seen between the structure of patterns and the syntax or grammar of languages.

- Following this analogy, patterns can be built up from sub-patterns in a number of ways, similarly to how one builds words by concatenating characters, and builds a phrase or sentence by concatenating words.
Definitions

- The simplest sub-patterns are called *pattern primitives*, and should be much easier to recognize than the overall patterns.
- The language used to describe the structure of the patterns in terms of sets of pattern primitives is called the *pattern description language*.
- The pattern description language will have a *grammar* that specifies how primitives can be composed into patterns.
Syntax Analysis

- When a primitive within the pattern is identified, syntax analysis (parsing) is performed on the sentence describing the pattern to determine if it is correct with respect to the grammar.

- Syntax analysis also gives a structural description of the sentence associated with the pattern.
Syntax Analysis

- One advantage of this approach is that a grammar (rewriting) rule can be applied many times.
- This allows for expressing basic structural characteristics for an infinite number of sentences in a number of compact ways.
Other Representations

- Relational graph - describe a pattern using the relations between sub-patterns and primitives.
- Relational matrix - any relational graph can also be expressed as a matrix.
Other Representations

- Generalizing to allow for any relation that can be determined from the pattern, we can express richer descriptions than through tree-based structures.

- Hierarchical (tree-based) approaches are convenient because it is easy to apply formal language theory.
Syntactic System

- Consists of two main parts:
  - Analysis - primitive selection and grammatical or structural inference
  - Recognition - preprocessing, segmentation or decomposition, primitive and relation recognition, and syntax analysis

- Preprocessing includes the tasks of pattern encoding and approximation, filtering, restoration, and enhancement.
Figure 1.10. Block diagram of a syntactic pattern recognition system.
Syntactic System

Figure 1: Using SyntPR for classification (with explicit characterization of structure).
Syntactic System

- After preprocessing, the pattern is segmented into sub-patterns and primitives using predefined operations.
- Sub-patterns are identified with a given set of primitives, so each pattern is represented by a set of primitives with the specified syntactic operations.
Syntax Parsing

- For example, using the concatenation operation, each pattern is recognized by a string of concatenated primitives.

- At this point, the parser will determine if the pattern is syntactically correct.
  - It belongs to the class of patterns described by the grammar if it is correct.
Syntax Parsing

- During parsing/syntax analysis, a description is produced in terms of a parse tree, assuming the pattern is syntactically correct.

- If it isn’t correct, it will either be rejected or analyzed based on a different grammar, which could represent other possible pattern classes.
Matching

- The simplest form of recognition is template matching, in which a string of primitives representing an input pattern is compared to strings of primitives representing reference patterns.

- The input pattern is classified in the same class as the prototype that is the best match, which is determined by a similarity criterion.
Matching vs. Complete Parsing

- In this case, the structural description is ignored.
- The opposite approach is a complete parsing that uses the entire structural description.
- There are many intermediate approaches; for example, a series of tests designed to test the occurrence of certain primitives, sub-patterns, or combinations of these. The result of these tests will determine a classification.
Parsing

- Parsing is required if the problem necessitates using a complete pattern description for recognition.
- Efficiency of the recognition process is improved by simpler approaches that do not require a complete parsing.
- Basically, parsing can be expensive, so don’t use it unnecessarily.
Inferring Grammars

- Grammatical inference machine - similar to “learning” in the discriminant approach; it infers a grammar from a set of training patterns.

- The inferred grammar can then be used for pattern description and syntax analysis.
Parsing - Fundamentals

- Parser Hierarchical Structure
  - Smaller decompositions
  - Graphically shown by derivation trees

```
the program crashes
```

```
the computer
```
Parsing Problems

- Approaches of Parsing
- Parsing/Generation Similarities
  - Application of grammar is easier in generative mode than analytic mode.
- Concerns
  - Parser must determine the extent of the elements that comprise non terminals.
  - Parser must find a use for all of x
Parsing Approaches

- **Top-Down Parsing**
  - From S to terminals. A derivation for x, where x is a sentence.
  - Method 1: Depth First Expansion of non-terminals, starting with leftmost non-terminal. Allows back-up.
  - Method 2: Recursive Descent may not work on all grammars. No back-up. Recursive functions to recognize sub-strings corresponding to the expansion of a non-terminal.

- **Bottom-Up Parsing**
  - Knowing x, we proceed to S by reversing the productions defined.
Comparing Top-down and Bottom-up

- Difficult to compare because the efficiency factor lies with the grammar.
- Normalization or Transformation of a grammar will affect parsing efficiency.
- Brute force method of the top-down and bottom-up approaches have computational complexity growing exponentially with $|x|$. 
Alternative Approaches – CYK Parsing

- Cocke-Younger-Kasami Algorithm
  - Parse string $x$ in number of steps proportional to $|x|^3$.
  - The CFG should be in Chomsky Normal Form
  - Building CYK table
CYK Parsing contd.

- The cell (1,n) should have S. Then the parsing is said to be complete.

- Example
  - Productions
    - $S \rightarrow AB|BB$
    - $B \rightarrow BB|CA|b$
    - $A \rightarrow CC|AB|\alpha$
    - $C \rightarrow BA|AA|b$

  - CYK table
Stochastic Grammars

- Assumptions of the formal grammar used in SyntPR
  - Languages are disjoint
  - No errors in the sentences produced by the grammar

- In practice the assumptions are faulty
  - Errors in the primitive extraction process
  - Noise or pattern deformation in descriptions
Stochastic Grammars contd.

- **Definition**
  - $G_s = \{V_N, V_T, P_s, S_s\}$
  - $P_s$ is a set of Stochastic Productions
    - Each production is of form $a_i \rightarrow b_j$ with probability $p_{ij}$

- **Derivations in Stochastic Language**
  - Derivations of sentence from $S_s$ to $x$
  - Labels $t_{k-1,k}$ where $k=1$ to $n$ are given to each production such as $\beta_{k-1}$ to $\beta_k$
  - Every production will have a probability $p_i$
  - Unconditional Probability is given by
    - $P(t_{0,1}', t_{1,2}', \ldots, t_{n-1,n}') = P(t_{0,1})P(t_{1,2})\ldots P(t_{n-1,n})$
Stochastic Grammars contd.

- \( P(t_{0,1}, t_{1,2}, \ldots, t_{n-1,n}) = \prod_{q=1}^{n} P(t_{q-1,q}) \)
- This uses the assumption that every production is independent of the previous one applied.

- Proper Stochastic Grammar
  - Elements of Ps is of form
    - \( A_i \rightarrow \beta_i \) with probability \( p_{ij} \)
    - Where \( A_i \in V_N, \beta_i \in (V_N \cup V_T)^+ \)
    - \( \sum_{k=1}^{n_i} p_{ik} = 1 \) (Sum of all the probabilities of each production in the Grammar is equal to 1)
Stochastic Grammars contd.

- **Characteristic Grammar**
  - Remove the probability measure from the Stochastic grammar

- **Stochastic Languages**
  - \( L(G_S) = \{(x,p(x)) \mid x \in V_T^+, S_S \text{ derives } x \text{ with probability } p_j, j = 1 \text{ to } k, p(x) = \sum_{j=1}^{k} p_j \} \)
  - Where \( p_j \) is the probability to parse a string \( x \) from \( S_S \) and \( p(x) \) is the total probability of deriving various strings (Say \( k \) number of strings) using the grammar.
Stochastic Grammars contd.

- For example, x is ‘abc’ and productions of a grammar are:
  - S->aA with $p_1$; A->bC with $p_2$
  - B->dC with $p_3$; C->eD with $p_4$
  - B->c with $p_5$; B->f with $p_6$
  - B->g with $p_7$; C->c with $p_8$
  - C->f with $p_9$; C->g with $p_{10}$
  - D->c with $p_{11}$; D->f with $p_{12}$
  - D->g with $p_{13}$

- Then to get x we have S->aA->abC->abc.
- Here the probability to get abc is $p(abc)=p_1.p_2.p_8$
- $p_1+p_2+...+p_{13} = 1$ if the given grammar is Proper Stochastic Grammar
Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation

Paper by Roberto Navigli and Paola Verlardi
Word-Sense Disambiguation

- Same word, different meaning. For example, “bus” can be a vehicle or a connection on a computer.
- This leads to ambiguous situations in which it is not clear which word to use.
- This paper’s approach uses syntactic pattern recognition in attempting to improve disambiguation.
Representation

- Used a graph representation of senses:
Data

- Took data from a number of sources:
  - WordNet 2.0 - online resource featuring concepts that correspond to word senses
  - Domain labels assigned to WordNet
  - Annotated corpora - text examples of word sense usages in context
  - Dictionaries of collocations - words that belong to a semantic domain (i.e.: bus, stop, station)
Algorithm

- $T = [t_1, \ldots, t_n], I = [St_1, \ldots St_n], P = \{t_i | S^{t_i} = null\}$

- Algorithm iteratively disambiguates words in the pending set $P$ of words that have no currently defined sense, where $S$ is the chosen sense for $t$. 
Grammar

- Describes meaningful connections in the graph representation.
- Used to do the disambiguation task in the iterative algorithm.
Results

- Performed better on large contexts.
- Achieved a 66% recall rate when the number of elements in T is 5.
- Achieved around a 90% recall rate where the number of elements in T is 40.
Synt Pattern Recognition of ECG

- Trahanias, P and Skordalakis, E speak about how to recognize ECG information using SyntPR
- Patterns and Pattern parameters
- Primitive pattern selection
- Pattern Grammar
- Experimental results are convincing
References

- Fu, K. S. (King Sun), 1930- “Syntactic pattern recognition and applications” Englewood Cliffs, N.J. : Prentice-Hall, c1982
Thanks!