Syntactic Pattern Recognition

Anna Kuchna
Maciej Żarnowski

Wprowadzenie

- Pattern recognition (rozpoznawanie wzorców) jest gałęzią sztucznej inteligencji zajmującą się klasyfikacją i opisem obserwowanych obiektów.
- Celem rozpoznawania wzorców jest klasyfikacja danych, bazująca albo na wcześniejszej wiedzy o nich, albo na statystycznych informacjach uzyskanych z wzorców.

System rozpoznawania wzorców

- Mechanizm ekstrakcji charakterystycznych cech obserwowanego obiektu (składowe prymitywne) wyznaczający ich numeryczną albo symboliczną reprezentację (feature extraction mechanism)
- Schemat klasyfikacji i opisu

Różne podejścia do klasyfikacji i opisu rozpoznanych wzorców

- Statystyczne – bazujące na statystycznych charakterystykach wzorca
- Strukturalne – dekompozycja obiektu na składowe pierwotne, analiza relacji zachodzących pomiędzy nimi
- Syntaktyczne – polega na znalezieniu formalnego opisu obserwowanych obiektów

Schemat

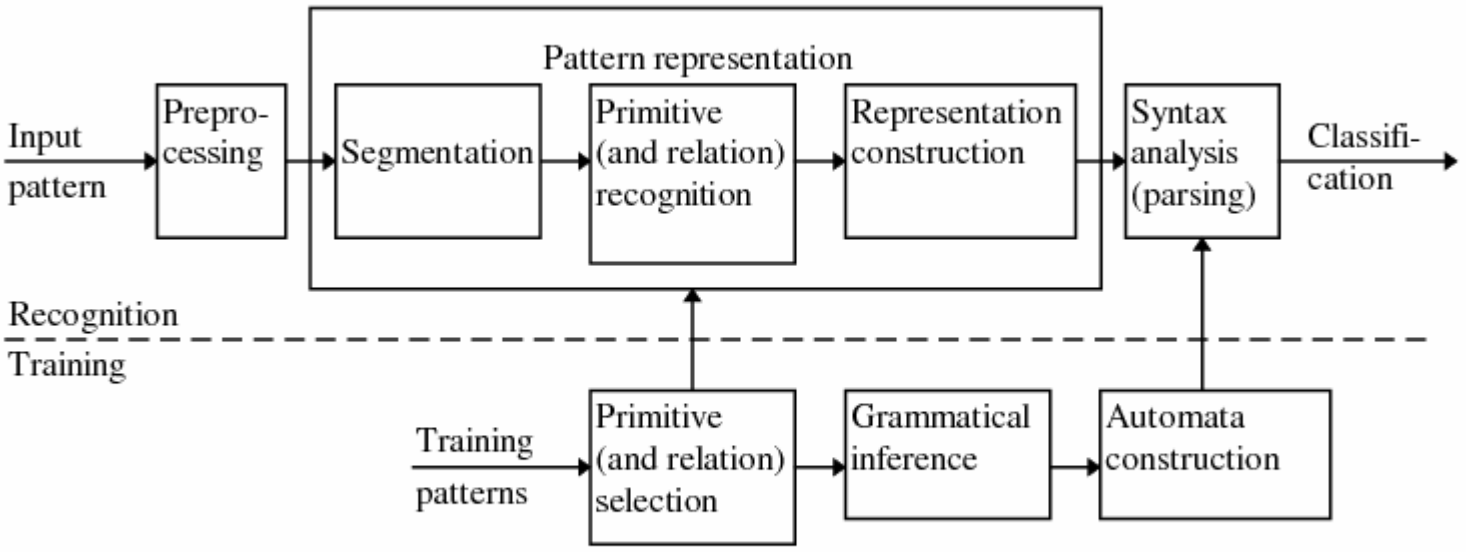


Fig. 1.1. Block diagram of a syntactic pattern recognition system.

Idea

- Obiekt opisujemy za pomocą słowa, trzeba sprawdzić, czy należy do pewnego języka
- Definiuje się produkcje, gramatyki i rozpoznające słowa automaty skończone

Zastosowania

- rozpoznawanie znaków alfabetu angielskiego i chińskiego (pismo odręczne)
- rozpoznawanie odcisków palców
- rozpoznanie mowy
- analiza trajektorii cząstek elementarnych w komorze pęcherzykowej (Wilsona),
- analiza danych biomedycznych (obrazy chromosomów, EEG)
- rozpoznawanie struktur chemicznych
- analiza sygnałów sejsmicznych etc.

Przykład zastosowania: rozpoznawanie obrazu

Rodzaje podejść syntaktycznych:

- Ciągowe
  - kody łańcuchowe Freemana
  - język opisu obrazu
- Drzewowe
  - T – drzewa proste (bez etykiet i skierowania)
  - EDT – edge-labeled directed tree
- Grafowe
  - Wykorzystywane raczej do opisu obrazów, mniej do rozpoznawania

Syntaktyczne rozpoznawanie ciągów

- Gramatyka generująca ciągi (*string grammar*):

gdzie:

N - skończony zbiór symboli nieterminalnych,

Σ - skończony zbiór symboli terminalnych,

P - zbiór produkcji,

$S \in N$ - symbol początkowy

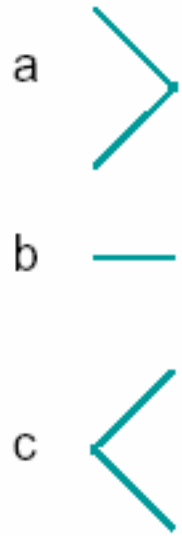
Notacja: V* - zbiór wszystkich ciągów (zdań) złożonych z symboli z V

Gramatyki

Gramatyki wykorzystywane szczególnie często w rozpoznawaniu obrazów:

- regularne
- bezkontekstowe

Przykład: rozpoznawanie obrazu



$$N = \{A, B, S\}$$

$$\Sigma = \{a, b, c\}$$

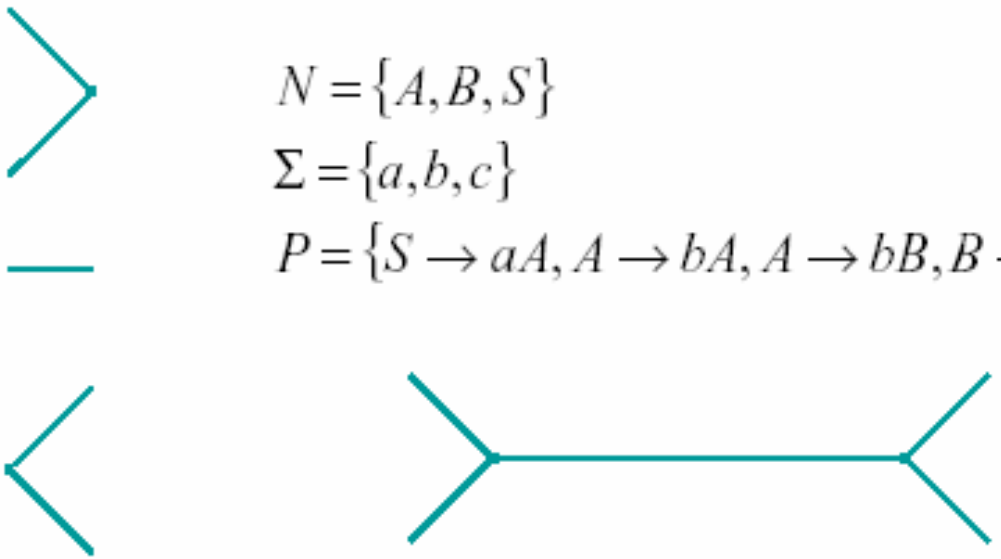
$$P = \{S \to aA, A \to bA, A \to bB, B \to c\}$$

Przykład cd.

a

b

c



$$N = \{A, B, S\}$$

$$\Sigma = \{a, b, c\}$$

$$P = \{S \to aA, A \to bA, A \to bB, B \to c\}$$

Wywód: S⇒aA⇒abA⇒abbA⇒...⇒abbbbbA ⇒abbbbbc

⇒ - krok wywodu

$$L(G) = \{ab^n c \mid n \geq 1\}$$

Semantyka

Czasami dogodnie jest stworzyć bazę danych zawierającą *reguły semantyczne*, formułujące pewne warunki narzucane na składowe pierwotne (SP), w tym:

- reguły dotyczące dopuszczalnych sposobów łączenia SP (np. łączenie tylko na końcach),
- wielkość SP,
- orientacja SP,
- ograniczenia na liczbę użyć danej produkcji w wywodzie (<, >).

Semantyka - przykład

Reguły semantyczne użyte niejawnie w wywodzie w poprzednim przykładzie:

- 1) połączenia tylko w punktach; kierunek a = dwusieczna kata wyznaczonego “ramionami”; długość 3cm
- 2) połączenia tylko w punktach; wielokrotne połączenia są niedozwolone; kierunek b musi być taki sam jak kierunek a; długość b 0.25cm; wolno stosować co najwyżej 10 razy
- 3) kierunki a i b musza być zgodne; połączenia tylko w punktach;

Gramatyki a automaty

- Zachodzi wzajemnie jednoznaczna odpowiedniość pomiędzy gramatykami regularnymi a automatami skończonymi.
- Dla każdej gramatyki regularnej G da się zbudować automat skończony który akceptuje jedynie słowa z L(G).

Automat skończony:

$$A = (Q, \Sigma, \delta, F, q, d)$$

gdzie:

- Q - skończony niepusty zbiór stanów,
- Σ - skończony alfabet wejściowy,
- δ - mapowanie: $Q \times \Sigma \rightarrow 2Q$
- q0 - stan początkowy,
- $F \subset Q$ - zbiór stanów końcowych (akceptujących)

Gramatyki drzewowe,syntaktyczne rozpoznawanie drzew

- Gramatyki drzewowe są bardzo przydatne w reprezentacji bardziej skomplikowanych wzorców 2-D lub więcej wymiarowych (tj. takich, które składają się z składników prymitywnych połączonych w kilku miejscach

Gramatyka generująca drzewa (*tree grammar*):

$$G = (N, \Sigma, P, r, S)$$

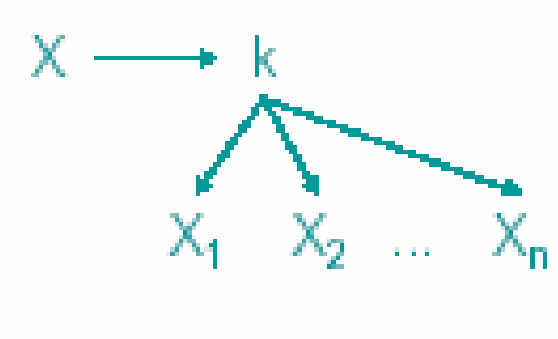
gdzie:

- N - skończony zbiór symboli nieterminalnych,
- Σ - skończony zbiór symboli terminalnych,
- P - zbiór produkcji; produkcje mają postać:

  $Ti \rightarrow Tj$, gdzie Ti i Tj są drzewami,
- r - funkcja rangująca (*ranking function*), podająca liczbę bezpośrednich potomków terminala
- $S \in N$ - symbol początkowy (może być drzewem)

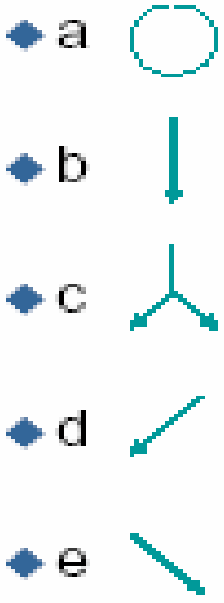
Syntaktyczne rozpoznawanie drzew

- Szczególny przypadek, najczęściej wykorzystywany w rozpoznawaniu obrazów: drzewowe gramatyki ekspansywne (*expansive tree grammars*), w których wszystkie produkcje są postaci:



Przykład

- a
- b
- c
- d
- e



$S \rightarrow a\ X_1$

$X_1 \rightarrow b$

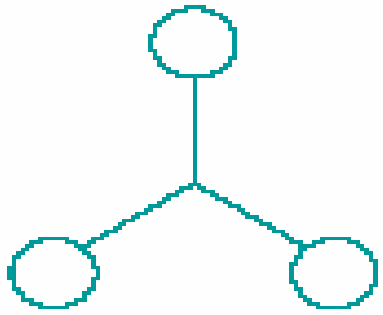
$X_1 \rightarrow c\ X_2\ X_3$

$X_2 \rightarrow d$

$X_2 \rightarrow a$

$X_3 \rightarrow e$

$X_3 \rightarrow a$



Zalozenia:

- polaczenia od glowy do ogona,
- dla *a*: dowolnie (na obwodzie)

Uczenie w podejściach syntaktycznych

- Jawne wyspecyfikowanie gramatyki przez projektanta systemu jest zazwyczaj nierealne. Pożądane: możliwość uczenia się (konstruowania) automatów na podstawie przykładowych wzorców (ciągów, drzew, etc.).
- Ponieważ zachodzi odpowiedniość automatów i gramatyk, problem ten sprowadza się do problemu uczenia się gramatyk z przykładów (tzw. wywodzenie gramatyk, *grammatical inference*).

Uczenie w podejściach syntaktycznych c.d.

Załóżmy ze

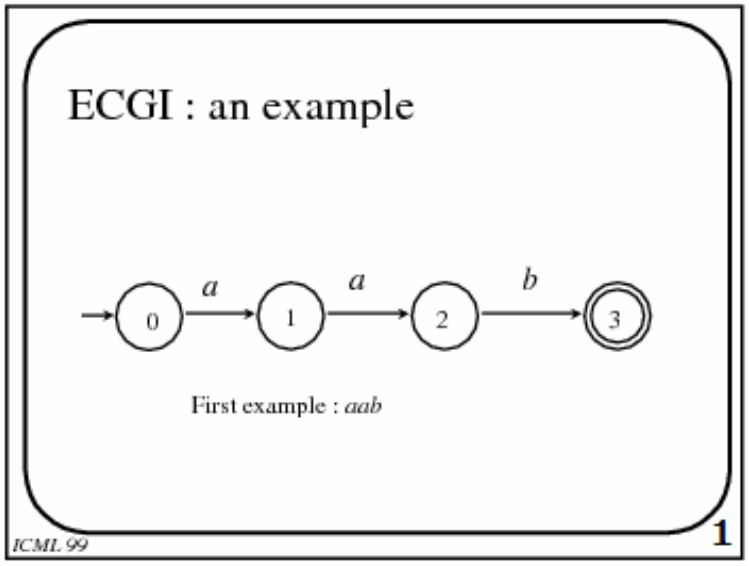
- wszystkie wzorce należące do klasy, która chcemy rozpoznawać, generowane są przez nieznana gramatykę G,
- mamy do dyspozycji skończony zbiór uczący (przykładów pozytywnych).

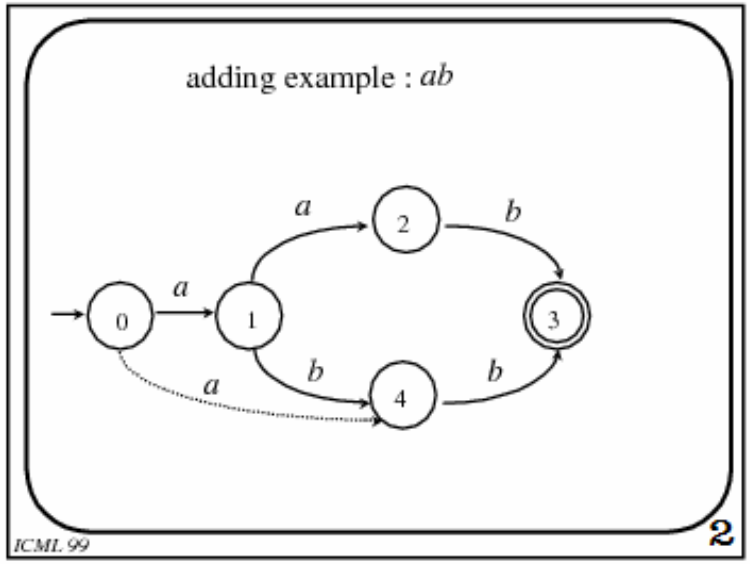
$$R+ \subseteq \{a \mid a \in L(G)\}$$

- O zbiorze R+ powiemy ze jest *strukturalnie kompletny* jeżeli każda produkcja z G jest wykorzystywana do wygenerowania przynajmniej jednego przykładu uczącego z R+.
- Cel: synteza skończonego automatu Af który będzie akceptował ciągi z R+ oraz (być może) pewne ciągi *podobne* do R+ (indukcja!).

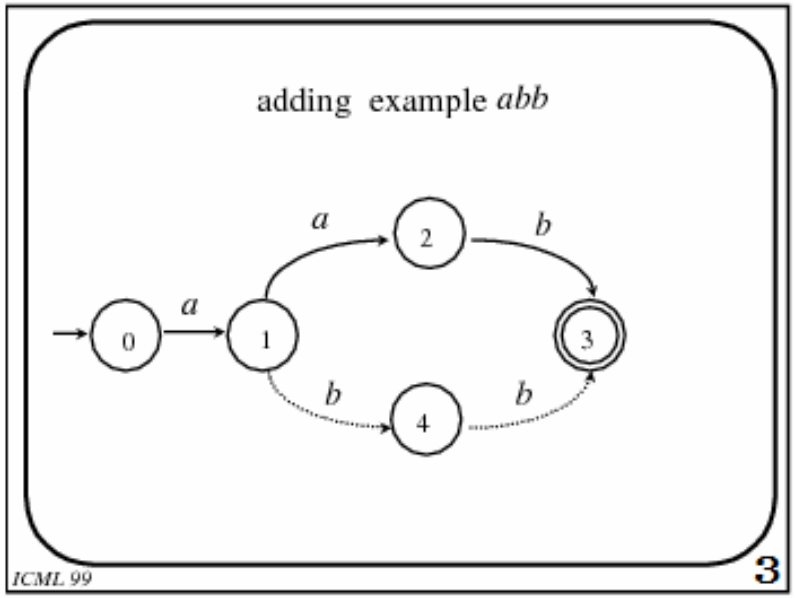
Algorytmy wywodzenia gramatyk

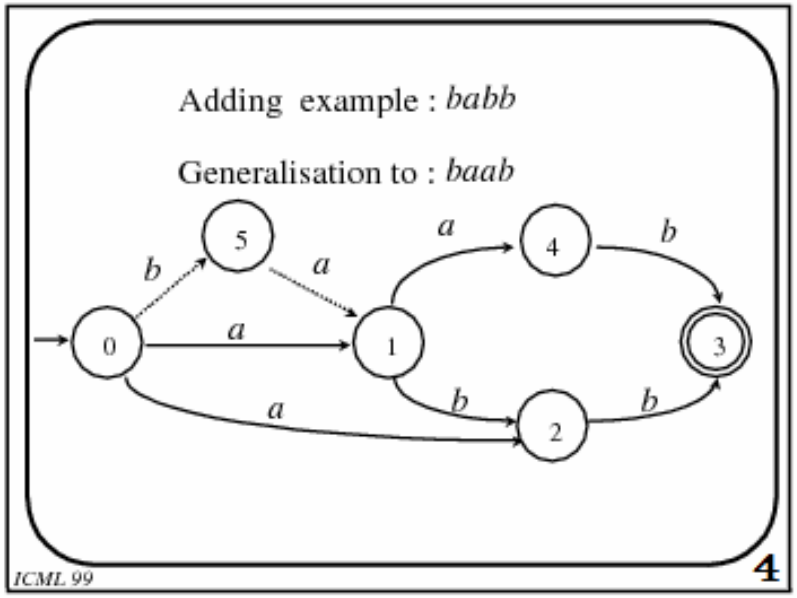
- ECGI (error-correction grammatical inference) – oparty na koncepcji różnic pomiędzy łańcuchami





Przykład ECGI cd.

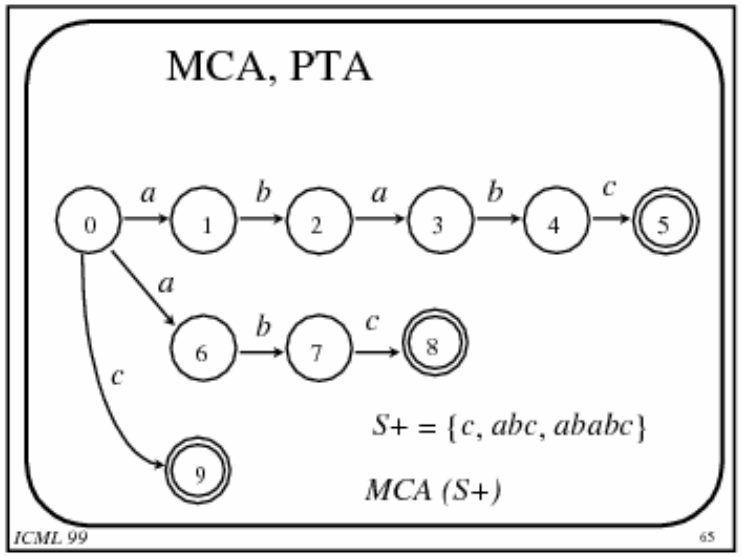


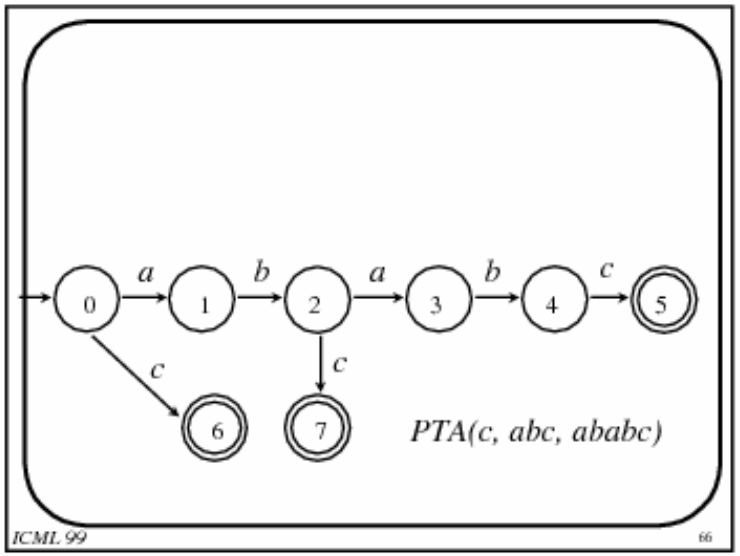


Algorytmy wywodzenia gramatyk c.d.

- MCA (maximal canonical automaton)
- PTA (prefix tree acceptor)

  negatywne wzorce pomagają usuwać nadmierna generalizacje





Algorytmy wywodzenia gramatyk c.d.

- Algorytmy genetyczne – schemat działania:
  1. Zakoduj automat jako string
  2. Zdefiniuj crossover
  3. Zdefiniuj optimum jako automat z najmniejsza liczba stanów, nie akceptujący przykładów negatywnych
  4. Wykonuj algorytm genetyczny do osiągnięcia warunku stopu

Wady i zalety podejścia syntaktycznego

Zalety:

- elegancki formalizm
- podejście ‘całościowe’

Wady i zalety podejścia syntaktycznego

Wady:

- problemy z uwzględnianiem zaszumienia danych (obrazów) i danymi brakującymi,
- problemy z automatycznym generowaniem gramatyk z przykładów,
- problemy z aktualizacja gramatyki/automatu gdy pojawiają się nowe przykłady.
- złożoność obliczeniowa analizy syntaktycznej: NP (dla większości klas gramatyk)

Ciekawe odnośniki

- Optical Character Recognition:

  http://www.dsic.upv.es/users/tlcc/papers/fullpapers/SL03.pdf
- Face Recognition

  http://www.face-rec.org/interesting-papers/General/zhao00face.pdf