Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №5

Мультипарадигменне програмування

Виконав
студент групи ІА-23:
Курач В.А.

Київ 2025

Завдання: на процедурній мові програмування реалізувати перетворення чисельного ряду до лінгвістичного ланцюжка за певним розподілом ймовірностей потрапляння значень до інтервалів з подальшою побудовою матриці передування.

Вхідні дані: чисельний ряд, вид розподілу ймовірностей, потужність алфавіту.

Вихідні дані: лінгвістичний ряд та матриця передування.

1. Постановка задачі

Метою лабораторної роботи було створення програми в середовищі **CLIPS**, що виконує перетворення числового часового ряду у лінгвістичний ряд. Такий підхід дозволяє представити числові дані у вигляді послідовності символів (лінгвістичний ряд), що може бути використано для подальшого аналізу, наприклад, у задачах класифікації, аналізу тенденцій тощо.

Для досягнення мети необхідно реалізувати:

- Зчитування числових даних з CSV-файлу.
- Дискретизацію значень на інтервали згідно з заданим алфавітом.
- Побудову матриці переходів між символами алфавіту на основі отриманого лінгвістичного ряду.

Для перевірки роботи використовувалися часові ряди з попередніх лабораторних робіт, що дозволяє провести порівняльний аналіз.

2. Хід роботи та розв’язання задачі

Розв’язання задачі було реалізовано шляхом написання низки **функцій CLIPS**, які послідовно виконують необхідні кроки:

1. **Зчитування CSV-файлу**:
    a. Реалізована функція read-csv-column, яка відкриває файл, зчитує заголовки, знаходить індекс стовпця з назвою "Price", і формує список числових значень.
2. **Формування алфавіту**:
    a. Якщо користувач не надає конкретний алфавіт, використовується автоматично згенерований набір символів латинського алфавіту (generate-alphabet) заданої довжини.

3. **Розбиття на інтервали**:
    a. Значення часового ряду розбиваються на рівномірні інтервали відповідно до потужності алфавіту (get-intervals-uniform).
4. **Мапінг значень на символи**:
    a. Кожне значення зі списку чисел відображається на символ відповідно до інтервалу, у який воно потрапляє (map-to-alphabet).
5. **Формування матриці переходів**:
    a. Побудована матриця частот переходів між символами (build-transition-matrix), де кожен елемент [i][j] означає кількість переходів від символу i до символу j.
6. **Вивід результатів**:
    a. Символьний ряд та матриця переходів виводяться у зручному для аналізу вигляді (print-matrix).

Розробка реалізована згідно з принципами **функціонального програмування**: відсутність глобальних змінних, чисті функції, передача параметрів явно.

3. Використані інструменти та середовище

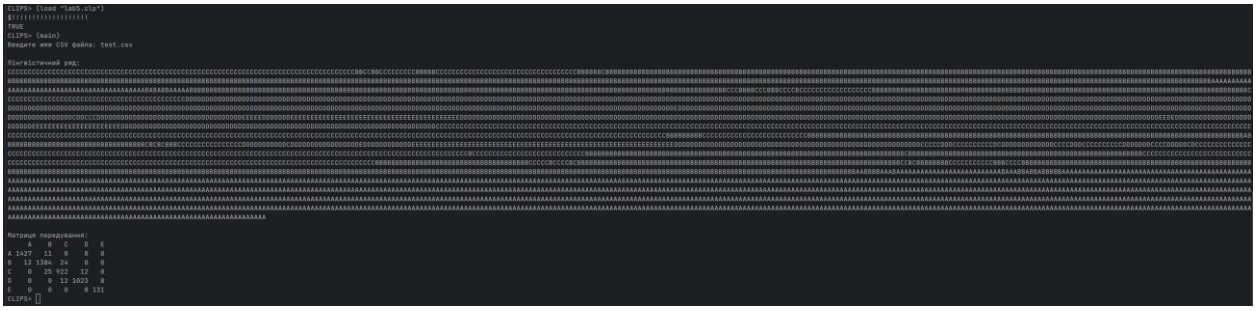
- **CLIPS** — середовище для експертних систем, яке підтримує декларативне та функціональне програмування.
    - Вбудовані інструменти CLIPS:
        - readline, open, str-explode, sub-string, create$, foreach, eval, str-cat, printout тощо.
    - CSV-файли, які містять числові часові ряди, зокрема значення у стовпці Price.

4. Результати роботи

Результатом виконання програми є:

- **Лінгвістичний ряд**, у якому кожне значення числового часового ряду відображене у відповідний символ алфавіту.

- **Матриця переходів**, яка наочно демонструє частоту переходів між символами у часі, що може бути використано для подальшого статистичного аналізу або в задачах класифікації поведінки.



Лістинг програми:

```
(deffacts start
  (file-opened FALSE)
  (line-number 0)
  (prices-list (create$))
  (alphabet (create$))
  (breaks (create$))
  (mapped-series (create$))
  (matrix (create$))
)

(deffunction read-line-from-file (?file)
  (bind ?line (readline ?file))
  (if (eq ?line EOF) then FALSE else ?line)
)

(deffunction str-explode (?str)
  (bind ?result (create$))
  (bind ?temp "")
  (bind ?len (str-length ?str))
  (bind ?i 1)
  (while (<= ?i ?len) do
    (bind ?c (sub-string ?i ?i ?str))
    (if (eq ?c ",") then
      (if (neq ?temp "") then (bind ?result (create$ ?result ?temp)))
      (bind ?temp "")
    else
      (bind ?temp (str-cat ?temp ?c))
    )
    (bind ?i (+ ?i 1))
  )
  (if (neq ?temp "") then (bind ?result (create$ ?result ?temp)))
  ?result
)

(deffunction find-price-index (?headers)
  (bind ?len (length$ ?headers))
  (bind ?idx 1)
```

```
 (while (<= ?idx ?len) do
  (if (eq (nth$ ?idx ?headers) "Price") then (return ?idx))
  (bind ?idx (+ ?idx 1))
 )
 0
)

(deffunction my-str-replace (?str ?old ?new)
 (bind ?pos (str-index ?old ?str))
 (while (neq ?pos 0) do
  (bind ?str (str-cat
             (if (> ?pos 1) then (sub-string 1 (- ?pos 1) ?str) else "")
             ?new
             (sub-string (+ ?pos (str-length ?old)) (str-length ?str) ?str)))
  (bind ?pos (str-index ?old ?str))
 )
 ?str
)

(deffunction str-to-float (?str)
 (eval ?str)
)

(deffunction read-csv-column (?filename)
 (if (open ?filename "r" "file1") then
  (bind ?file "file1")
  (bind ?headerLine (read-line-from-file ?file))
  (if (eq ?headerLine FALSE) then
   (close ?file)
   FALSE
  else
   (bind ?headers (str-explode ?headerLine))
   (bind ?priceIndex (find-price-index ?headers))
   (if (eq ?priceIndex 0) then
    (close ?file)
    FALSE
   else
    (bind ?prices (create$))
    (bind ?line (read-line-from-file ?file))
    (while (neq ?line FALSE) do
     (bind ?fields (str-explode ?line))
     (if (>= (length$ ?fields) ?priceIndex) then
      (bind ?priceStr (nth$ ?priceIndex ?fields))
      (bind ?cleanPrice (my-str-replace ?priceStr "," ""))
      (bind ?num (str-to-float ?cleanPrice))
      (if (neq ?num FALSE) then
       (bind ?prices (create$ ?prices ?num))
      )
     )
     (bind ?line (read-line-from-file ?file))
    )
    (close ?file)
    ?prices
   )
  )
 else
  FALSE
 )
)
```

```
(deffunction min-list (?lst)
  (bind ?min (nth$ 1 ?lst))
  (foreach ?x ?lst
    (if (< ?x ?min) then (bind ?min ?x))
  )
  ?min
)

(deffunction max-list (?lst)
  (bind ?max (nth$ 1 ?lst))
  (foreach ?x ?lst
    (if (> ?x ?max) then (bind ?max ?x))
  )
  ?max
)

(deffunction generate-alphabet (?size)
  (bind ?alphabet (create$))
  (bind ?letters (create$ "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y"
"Z"))
  (bind ?i 1)
  (while (<= ?i ?size) do
    (bind ?alphabet (create$ ?alphabet (nth$ ?i ?letters)))
    (bind ?i (+ ?i 1))
  )
  ?alphabet
)

(deffunction get-intervals-uniform (?series ?alphaSize)
  (bind ?min (min-list ?series))
  (bind ?max (max-list ?series))
  (if (eq ?min ?max) then
    (bind ?breaks (create$))
    (bind ?i 0)
    (while (<= ?i ?alphaSize) do
      (bind ?breaks (create$ ?breaks ?min))
      (bind ?i (+ ?i 1))
    )
    ?breaks
  else
    (bind ?step (/ (- ?max ?min) ?alphaSize))
    (bind ?breaks (create$))
    (bind ?i 0)
    (while (<= ?i ?alphaSize) do
      (bind ?b (+ ?min (* ?i ?step)))
      (bind ?breaks (create$ ?breaks ?b))
      (bind ?i (+ ?i 1))
    )
    ?breaks
  )
)

(deffunction find-interval (?value ?breaks ?alphabet)
  (bind ?len (length$ ?breaks))
  (bind ?i 1)
  (while (< ?i ?len) do
    (bind ?low (nth$ ?i ?breaks))
    (bind ?high (nth$ (+ ?i 1) ?breaks))
    (if (and (>= ?value ?low) (< ?value ?high)) then
      (return (nth$ ?i ?alphabet))
```

```
   )
   (bind ?i (+ ?i 1))
  )
  (nth$ (- ?len 1) ?alphabet)
)

(deffunction map-to-alphabet (?values ?breaks ?alphabet)
  (bind ?mapped (create$))
  (foreach ?v ?values
   (bind ?ch (find-interval ?v ?breaks ?alphabet))
   (bind ?mapped (create$ ?mapped ?ch))
  )
  ?mapped
)

(deffunction init-matrix (?size)
  (bind ?matrix (create$))
  (bind ?i 1)
  (while (<= ?i ?size) do
   (bind ?row (create$))
   (bind ?j 1)
   (while (<= ?j ?size) do
    (bind ?row (create$ ?row 0))
    (bind ?j (+ ?j 1))
   )
   (bind ?matrix (create$ ?matrix ?row))
   (bind ?i (+ ?i 1))
  )
  ?matrix
)

(deffunction replace-nth (?list ?index ?value)
  (bind ?result (create$))
  (bind ?len (length$ ?list))
  (bind ?i 1)
  (while (<= ?i ?len) do
   (if (eq ?i ?index) then
    (bind ?result (create$ ?result ?value))
   else
    (bind ?result (create$ ?result (nth$ ?i ?list)))
   )
   (bind ?i (+ ?i 1))
  )
  ?result
)

(deffunction update-matrix (?matrix ?row ?col)
  (bind ?oldRow (nth$ ?row ?matrix))
  (bind ?oldVal (nth$ ?col ?oldRow))
  (bind ?newRow (replace-nth ?oldRow ?col (+ ?oldVal 1)))
  (replace-nth ?matrix ?row ?newRow)
)

(deffunction index-of (?elem ?list)
  (bind ?len (length$ ?list))
  (bind ?i 1)
  (while (<= ?i ?len) do
   (if (eq (nth$ ?i ?list) ?elem) then (return ?i))
   (bind ?i (+ ?i 1))
  )
```

```
  FALSE
)

(deffunction build-transition-matrix (?mapped ?alphabet)
  (bind ?size (length$ ?alphabet))
  (bind ?matrix (init-matrix ?size))
  (bind ?len (length$ ?mapped))
  (bind ?i 1)
  (while (< ?i ?len) do
    (bind ?from (nth$ ?i ?mapped))
    (bind ?to (nth$ (+ ?i 1) ?mapped))
    (bind ?fromIdx (index-of ?from ?alphabet))
    (bind ?toIdx (index-of ?to ?alphabet))
    (if (and ?fromIdx ?toIdx) then
      (bind ?matrix (update-matrix ?matrix ?fromIdx ?toIdx))
    )
    (bind ?i (+ ?i 1))
  )
  ?matrix
)

(deffunction print-matrix (?matrix ?alphabet)
  (printout t "Transition matrix:" crlf)
  (printout t "    ")
  (foreach ?ch ?alphabet (printout t ?ch " "))
  (printout t crlf)
  (bind ?size (length$ ?alphabet))
  (bind ?i 1)
  (while (<= ?i ?size) do
    (printout t (nth$ ?i ?alphabet) " | ")
    (bind ?row (nth$ ?i ?matrix))
    (bind ?j 1)
    (while (<= ?j ?size) do
      (printout t (nth$ ?j ?row) " ")
      (bind ?j (+ ?j 1))
    )
    (printout t crlf)
    (bind ?i (+ ?i 1))
  )
)

(deffunction main ()
  (printout t "Enter CSV filename: ")
  (bind ?filename (readline))
  (bind ?prices (read-csv-column ?filename))
  (if (neq ?prices FALSE) then
    (bind ?alphaSize 5)
    (bind ?alphabet (generate-alphabet ?alphaSize))
    (bind ?breaks (get-intervals-uniform ?prices ?alphaSize))
    (bind ?mapped (map-to-alphabet ?prices ?breaks ?alphabet))
    (foreach ?ch ?mapped (printout t ?ch))
    (printout t crlf)
    (bind ?matrix (build-transition-matrix ?mapped ?alphabet))
    (print-matrix ?matrix ?alphabet)
  )
)
```

Висновки: У результаті виконання лабораторної роботи було реалізовано програму в середовищі CLIPS, яка здійснює перетворення числового часового ряду у лінгвістичний ряд та формує матрицю переходів між символами. Програма дозволяє зчитувати дані з CSV-файлу, розбивати числові значення на рівномірні інтервали відповідно до розміру заданого алфавіту та відображати кожне значення у відповідний символ. Побудована матриця переходів демонструє частоту переходів між символами у лінгвістичному ряді. Отримані результати підтвердили правильність реалізації алгоритму та його застосовність для аналізу часових рядів.