Metody eksploracji danych - projekt

Dokumentacja

Przemysław Barcikowski, Dariusz Dudziński

18 stycznia 2015

1 Zadanie

1.1 Temat

Wyszukiwanie uogólnionych wzorców sekwencyjnych (Generalized Sequential Patterns)

1.2 Cel projektu

Celem niniejszego projektu jest stworzenie aplikacji spełniającej poniższe wymagania:

- Aplikacja służy do badania sekwencyjnych reguł asocjacyjnych na podstawie notowań giełdowych,
- Aplikacja przyjmuje pliki z danymi w formacie .csv, gdzie pierwsza kolumna to nazwa serii, druga to data. Kolejne atrybuty będą wykorzystywane do samego wyliczania sekwencji. Są to wartości liczbowe,
- Aplikacja automatycznie buduje taksonomię na atrybutach liczbowych w postaci
  - zaokrąglenie do pełnej wartości,
  - wartość dodatnia/ujemna.
- Aplikacja operuje z zadanym przez użytkownika parametrami,
- Aplikacja generuje sekwencyjne reguły asocjacyjne,
- Aplikacja jest napisana w języku Java.

Dodatkowo:

- Aplikacja powinna zostać przetestowana zarówno pod kątem poprawności, jak i wydajności,
- należy przeprowadzić eksperymenty mające na celu wyszukanie ciekawych wzorców sekwencyjnych.

2 Rozwiązanie

Do rozwiązania wyżej postawionego problemu został wykorzystany algorytm Generalized Sequential Patterns (GSP) [2]. Jako dane do testowania oraz eksperymentów wybrano notowania giełdowe indeksu Dow Jones, pobrane ze strony [1].

3 Implementacja

3.1 Funkcjonalności aplikacji

Aplikacja wykonuje algorytm GSP (opisany w [2]), czyli wyszukuje wszystkie uogólnione wzorce sekwencyjne w zadanym pliku z danymi, przy zadanych parametrach algorytmu (patrz 3.2). Jako wynik działania, aplikacja kieruje do standardowego strumienia wyjścia następujące informacje:

- Raport z wyszukiwania wzorców sekwencyjnych (przykład: Listing 1.),

```
SEQUENCE SEARCH REPORT:

Step: 1
generated candidates :76
candidates rejected by hash tree: 64
confirmed sequences  :12
ver true 12
```

Listing 1: Raport z wyszukiwania

- Podsumowanie wykonania algorytmu (przykład: Listing 2.), zawierające:
  - Zadane parametry,
  - Informacje dotyczące samego wykonania algorytmu,
  - Wskaźniki wydajności.

```
SUMMARY

Parameters:
file:         testdata/test4.csv
minSupp:      2
minGap:       28
maxGap:       49
timeConstr:   365
widnowSize:   7
useHashTree:  true
hierarchy   : false

Execution info:
execTime: 478ms
Pattern Sequence found: 325
Pattern Sequence reduced by hash tree : 91
Longest: 7

Performance indicators:
Ratio [Confirmed Sequences/Generated Candidates]: 0.3722794959908362
Exec time per confirmed sequence: 1.4707692307692308ms
```

Listing 2: Podsumowanie

- Wyszukane sekwencje (przykład: Listing 3.).

```
RESULT SERIES:

support: 20 : close.sign:1 ,
support: 20 : close_change.sign:1 ,
support: 20 : volume.sign:1 ,
support: 20 : volume_change:0 ,
```

Listing 3: Wyszukane sekwencje

3.2 Konfiguracja parametrów programu

Parametry algorytmu są podawane za pomocą pliku konfiguracyjnego. Użytkownik może regulować następujące parametry algorytmu:

- wykorzystywanie drzewa hashującego (parametr $useHashTree$, zmienna binarna),
- wykorzystywanie taksonomii (parametr $useTaxonomies$, zmienna binarna),
- wielkość okna czasowego (parametr $slidingWindowSize$, zmienna całkowita, podawana w dniach),
- minimalne wsparcie (parametr $minSupport$, zmienna całkowita, podawana w dniach),
- minimalny odstęp (parametr $minGap$, zmienna całkowita, podawana w dniach),
- maksymalny odstęp (parametr $maxGap$, zmienna całkowita, podawana w dniach),
- ograniczenie czasowe (parametr $timeConstraint$, zmienna całkowita, podawana w dniach),
- ścieżka do pliku z danymi (parametr $dataFilePath$).

Przykładowy plik konfiguracyjny został pokazany na Listingu 4.

```
useHashTree=true
useTaxonomies=true
slidingWindowSize=7
minSupport=2
minGap=28
maxGap=49
timeConstraint=365
dataFilePath=testdata/test4.csv
```

Listing 4: Przykładowy plik konfiguracyjny

3.3 Zalecany sposób uruchamiania

Aplikację można uruchomić na kilka sposobów, zalecany to stworzenie wykonywalnego pliku *.jar* i wywoływanie go w linii poleceń. Możliwe opcje wywoływania:

- Z domyślnym plikiem konfiguracyjnym, jego nazwa to $config.properties$, musi się on znajdować w tej samej lokacji, co plik wykonywalny (przykład: Listing 5.)

```
java -jar programGSP.jar
```

Listing 5: Wywołanie dla domyślnego pliku konfiguracyjnego

- Z podaniem ścieżki do pliku konfiguracyjnego (przykład: Listing 6.)

```
java -jar programGSP.jar "config.properties_custom"
```

Listing 6: Wywołanie ze specyfikacją pliku konfiguracyjnego

W zależności od wielkości pliku z danymi, wynik działania programu może składać się z wielu linii tekstu. Z tego względu zaleca się przekierowanie wyniku programu do pliku tekstowego (przykład: Listing 7.)

```
java -jar programGSP.jar "config.properties_custom" > output.txt
```

Listing 7: Wywołanie dla domyślnego pliku konfiguracyjnego

3.4 Wybrana technologia i wydajność

Aplikacja została napisana w języku Java ze względu na łatwość implementacji. Zostało to okupione wydajnością aplikacji, gdyż Java nie należy do najszybszych języków programowania. Świadomie zrezygnowano z poprawy wydajności na rzecz wygody programowania ze względu na to, że aplikacja ma charakter jedynie demonstracyjny, nie została stworzona z myślą o zastosowaniu w biznesie ani badaniach naukowych.

4 Testy

4.1 Jakościowe

Poprawność działania aplikacji zostało przetestowane przy pomocy testów jednostkowych, zawartych w klasach $SequencePatternsTest$ oraz $CSVReaderTest$ (zawarte w plikach $.java$ o takich samych nazwach)

4.2 Wydajnościowe

Sprawdzono również wydajność programu, w zależności od wielkości okna czasowego (sliding window) oraz tego, czy zostały wykorzystane drzewo hashujące oraz taksonomia. Testy przeprowadzone dla dwóch plików z danymi: małego oraz dużego.

5 Eksperymenty i wyniki

Literatura

[1] Dow jones index data set. https://archive.ics.uci.edu/ml/datasets/Dow+Jones+Index. Accessed: 2015-01-17.

[2] Ramakrishnan Srikant ans Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements.