Metody eksploracji danych - projekt

Dokumentacja

Przemysław Barcikowski, Dariusz Dudziński

19 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]. Jest to algorytm posiadający następujące cechy:

- Wykorzystuje taksonomię,
- Pozwala na wykorzystanie okna czasowego(sliding window), oznacza to, że transakcje oddalone od siebie o czas mniejszy niż rozmiar okna, mogą być traktowane jako jedna transakcja,
- Wprowadza ograniczenia czasowe: minimalny(minGap) i maksymalny(maxGap) odstęp czasowy między transakcjami.

Jako dane do testowania oraz eksperymentów wybrano notowania giełdowe indeksu Dow Jones, pobrane ze strony [1].

3 Implementacja

3.1 Funkcjonalności aplikacji

3.1.1 Główne klasy

Podstawową funkcją klasy *CSVReader* jest budowa serii danych, na podstawie pliku csv. Plik wejściowy musi mieć w pierwszej kolumnie nazwę serii, a w drugiej datę. W trakcie przetwarzania tworzony jest słownik, który przypisuje unikalnym przedmiotom identyfikatory numeryczne, którymi operuje algorytm. Wykorzystywanie wartości numerycznych ułatwia i przyspiesza działanie.

Klasa *SequencePatterns* zawiera głowy kod aplikacji. Metoda *runAlgoritm*() działa w sposób iteracyjny, gdzie warunkiem stopu jest pusta lista kandydatów zweryfikowanych pod względem wsparcia. Metodę można uruchomić w dwóch trybach, z użyciem drzewa hashującego i bez. Pojedyncza iteracja zaczyna się od wygenerowania kandydatów zgodnie z zasadami algorytmu GSP, czyli z wykorzystaniem dwóch faz łączenia i przycinania. Następnie kandydaci są weryfikowani, w tym miejscu w zależności od trybu uruchomienia może być wykorzystanie drzewo hashujące, które ograniczy liczbę kandydatów do weryfikacji. Weryfikacja zgodna z algorytmem GSP analizuje serie danych z osobna, z wykorzystaniem faz poruszania się do przodu i do tyłu.

3.1.2 Dane wyjściowe

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.),

```
1 SEQUENCE SEARCH REPORT:
2
3 Step: 1
4 generated candidates :76
5 candidates rejected by hash tree: 64
6 confirmed sequences  :12
7 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 wsparcia. Testy przeprowadzone na pliku z danymi zawierającym 751 rekordów.

Wyniki algorytmu uruchomionego bez użycia taksonomii nie są interesujące, niezależnie od zestawu danych oraz innych parametrów. Algorytm znajdował bardzo mało sekwencji, wyniki nie nadawały się do żadnej sensownej interpretacji. Drzewo hashujące również było wykorzystywane za każdym razem, ze względu na istotność w przebiegu algorytmu. Następujące parametry były stałe dla wszystkich testów:

- $useHashTree$ = true,
- $useTaxonomies$ = true,
- $maxGap = 365$,
- $timeConstraint = 365$.

Tabela 1. przedstawia zestawy parametrów, które zmieniały się pomiędzy poszczególnymi testami.

Tabela 1: Przypadki testowe

Nr testu	$slidingWindowSize$	$minGap$	$minSupport$
1	7	14	20
2	14	28	20
3	7	14	25
4	14	28	25

Tabela 2. przedstawia wyniki testów. Przyjęto następujące oznaczenia dla wyników przedstawionych w Tabeli 2. :

potw/gen - stosunek potwierdzonych sekwencji do wygenerowanych kandydatów,

czas/sekw - śr. czas poświęcony na znalezienie potwierdzonej sekwencji,

odrzuceni - liczba wzorców sekwencyjnych odrzuconych przez drzewo hashujące w trakcie wykonywania algorytmu,

sekwencje - końcowa liczba znalezionych wzorców sekwencyjnych,

czas - czas wykonania algorytmu.

Tabela 2: Wyniki testów wydajnościowych

Nr testu	*potw/gen*	*czas/sekw* [ms]	*odrzuceni*	*sekwencje*	*czas* [s]
1	0.208	169.9	4039	2335	396.6
2	0.216	361.9	4019	2431	879.9
3	0.108	92.1	4019	699	64.4
4	0.174	191.0	4015	1406	268.6

Wyniki przedstawione w Tabeli 2. wykazują, że dla wyższego minimalnego wsparcia wykonanie algorytmu trwa krócej, a znalezionych wzorców jest mniej, co było do proste przewidzenia. Niebanalny wniosek jest taki, że dla mniejszego wsparcia (testy 1 i 2) efektywność wyszukiwania sekwencji niewiele się zmienia wraz z rozszerzeniem okna czasowego, natomiast ten parametr ma dość duże znaczenie przy większym wsparciu (testy 3 i 4). Dodatkowo, we wszystkich testach drzewo hashujące dla badanych kandydatów pozwoliło na znaczną redukcję liczby wzorców sekwencyjnych, które należało przeszukać w bezpośredni sposób.

5 Eksperymenty

5.1 Cel eksperymentów

Celem eksperymentów opisanych w niniejszej sekcji jest wyszukanie wzorców sekwencyjnych, które pomogą przewidywać zmiany cen akcji na giełdzie. Aby sekwencje nadawały się do wnioskowania, muszą mieć relatywnie wysokie wsparcie w tym przypadku max to 30). Dzięki temu będzie można mieć pewność, że wystąpienie znalezionego wzorca w przyszłości jest wysokie. Wnioskowanie nie powinno dotyczyć okna czasowego większego, niż miesiąc (większe okno może zbyt często dopuszczać niesłuszne traktowanie dwóch bardzo oddalonych od siebie transakcji jako jedną).

5.2 Dane

Oprogramowanie zostało napisane do badania notowań giełdowych, więc eksperymenty przeprowadzono na takich właśnie danych. Badano notowania indeksu Dow Jones, plik wsadowy zawierał 751 rekordów, dane pozyskano z [1].

5.3 Eksperymenty

5.3.1 Dla wąskiego okna czasowego

Listing 8. pokazuje parametry, dla których wykonano omawiany eksperyment.

```
useHashTree=true
useTaxonomies=true
slidingWindowSize=7
minSupport=20
minGap=14
maxGap=365
timeConstraint=365
```

Listing 8: Parametry, eksperyment 1

Spośród wzorców znalezionych dla parametrów podanych na Listingu 8. wybrano dwa najciekawsze, przedstawiono je na Listingu 9.

```
supp:24
0: {close_change.sign:1, volume_change.sign:-1},
1: {close_change.round:-3}

supp:20
0: {close_change.sign:-1, volume_change.sign:-1},
1: {close_change.round:2}
```

Listing 9: Wzorce, eksperyment 1

Z dwóch wzorców sekwencyjnych przedstawionych na Listingu 9. wynikają dwa analogiczne wnioski, które w pewien sposób potwierdzają się na wzajem. Pierwszy z nich oznacza, że dla 24/30 serii występuje następujący wzorzec: jeżeli zaobserwowano wzrost ceny akcji oraz spadek wolumenu obrotu giełdowego, to można się później spodziewać spadku cen akcji o ok. 3 procent. Drugi można wytłumaczyć tak, że dla 20/30 serii występuje wzorzec: jeżeli zaobserwowano spadek ceny akcji oraz spadek wolumenu obrotu giełdowego, to można się później spodziewać wzrostu cen akcji o ok. 2 procent.

5.3.2 Dla szerokiego okna czasowego

Listing 10. pokazuje parametry, dla których wykonano omawiany eksperyment.

```
useHashTree=true
useTaxonomies=true
slidingWindowSize=14
minSupport=20
minGap=28
maxGap=365
timeConstraint=365
```

Listing 10: Parametry, eksperyment 2

Ciekawa obserwacja: w przeciwieństwie do poprzedniego eksperymentu, w wynikach tego nie występują wzorce składające się z więcej, niż jednej sekwencji. Mimo tego, eksperyment dał kilka ciekawych wyników, np. sekwencje przedstawione na Listingu 11.

```
support: 20
0: {close_change.round:-4 , close_change.sign:-1 , volume_change.sign:1 , }
    ,

support: 20
0: {close_change.round:-4 , close_change.sign:-1 , volume_change.sign:-1 , }
     ,
```

Listing 11: Wzorce, eksperyment 2

Sekwencje przedstawione w Listingu 11 mają ten sam poziom wsparcia i różnią się miedzy sobą ostatnim elementem. Pierwszy wniosek jaki można stąd wyciągnąć brzmi następująco: obie sytuacje występują równie często (taki sam poziom wsparcia). Drugi wniosek: przedstawione sekwencje wzajemnie się wykluczają, ale jednak są poprawne. Powodem jest wykorzystanie okna czasowego i taksonomii (pełnej dla wszystkich elementów). Wykorzystywanie pełnej taksonomii i okna czasowego może prowadzić do wytworzenia wielu takich reguł, a użytkownik, który zasugeruje się takimi sekwencjami może na ich podstawie podjąć błędną decyzję.

5.3.3 Szukanie cotygodniowych sekwencji

Ten eksperyment to próba odnalezienia sekwencji, które zdarzają się tydzień po tygodniu. Listing 12. pokazuje parametry, dla których wykonano omawiany eksperyment.

```
useHashTree=true
useTaxonomies=true
slidingWindowSize=0
minSupport=15
minGap=7
maxGap=7
timeConstraint=365
```

Listing 12: Parametry, eksperyment 3

```
support: 15
0: {volume_change.sign:-1 , },
1: {close_change.sign:1 , } ,
2: {volume_change.sign:1 , } ,

support: 16
0: {volume_change.sign:-1 , }
1: {volume_change.sign:1 , }
2: {volume_change.sign:1 , }
```

Listing 13: Wzorce, eksperyment 3

Z wzorców pokazanych na Listingu 13. można wywnioskować, że po spadku wolumenu można się spodziewać jego wzrostu w 2 następnych tygodniach.

5.3.4 Szukanie max dwutygodniowych sekwencji

Ten eksperyment to powtórzenie wyżej opisanego z rozluźnieniem okna czasowego. Inaczej mówiąc, jest to próba sprawdzenia czego można się spodziewać w następnym, albo jeszcze kolejnym tygodniu po wystąpieniu danej sekwencji. Listing 14. pokazuje parametry, dla których wykonano omawiany eksperyment.

```
useHashTree=true
useTaxonomies=true
slidingWindowSize=0
minSupport=20
minGap=7
maxGap=14
timeConstraint=365
```

Listing 14: Parametry, eksperyment 4

```
support: 20
0: {close_change.round:2 , close_change.sign:1 , },
1: {volume_change.sign:1 , },
2: {volume_change.sign:-1 , },
3: {volume_change.sign:1 , },
4: {volume_change.sign:-1 , }

support: 25
0: {close_change.sign:1 , volume_change.sign:-1 , },
1: {volume_change.sign:1 , },
2: {volume_change.sign:-1 , },
3: {close_change.sign:1 , },
4: {volume_change.sign:1 , } ,

support: 20
0: {volume_change.sign:-1 , close_change.sign:-1 , },
1: {volume_change.sign:1 , },
2: {volume_change.sign:-1 , },
3: {volume_change.sign:1 , },
4 : {volume_change.sign:-1 , } ,
```

Listing 15: Wzorce, eksperyment 4

Ze wszystkich trzech wzorców znalezionych w tym eksperymencie można wyciągnąć jednakowy wniosek: serie początkowe mogą poprzedzać wahania wolumenu.

5.3.5 Poszukiwania bez użycia taksonomii

Taki eksperyment nie daje żadnych wyników, nawet przy dużym oknie czasowym. Należy więc pamiętać o tym, że korzystanie z taksonomii jest kluczowe przy wszystkich eksperymentach.

6 Wnioski

Algorytm jest mocno sparametryzowany, co z jednej strony jest korzyścią, jako że, potencjalnie można wygenerować więcej reguł, uwzględniając pewne rozluźnienie standardowego algorytmu generacji wzorców sekwencyjnych. Jednak nieodpowiedni dobór parametrów może prowadzić do wygenerowania błędnych wzorców. W szczególności gdy rozmiaru okna czasowego jest mniejszy lub równy dwukrotności minimalnego odstępu między transakcjami zachodzi sytuacja gdzie ten sam element jest brany pod uwagę wielokrotnie. W naszej aplikacji zastosowaliśmy dodatkowy warunek, który musi być spełniony żeby sekwencja została zaliczona. Warunek ma postać:

$$t_{min}(s_i) - t_{max}(s_{i-1}) \geqslant minGap \tag{1}$$

Wykorzystanie generycznych taksonomii w postaci

- zaokrąglenie do pełnej liczby,
- znak dodatni lub ujemny,

dla wartości numerycznych zaciemnił obraz, powstało dużo wzorców sekwencyjnych, które wydaje się, że nie wnoszą żadnej wiedzy.

Stworzona aplikacja spełnia wszystkie cele założone na początku projektu, działa poprawnie oraz jest wystarczająco wydajna, aby można było ją używać do prostych eksperymentów. Aplikacja w pełni nadaje się do demonstracji działania algorytmu GSP oraz celów dydaktycznych. Jej wydajność można oczywiście znacząco poprawić np. przepisując cały program na język C, lecz wymagało by to odrobinę więcej czasu, niż implementacja tych samych funkcjonalności w języku Java.

Stworzone oprogramowanie nie nadaje się do zastosowań badawczych ani biznesowych przez zbyt małą wydajność oraz niewielką liczbę funkcjonalności, ale stanowi dobrą podstawę do dalszego rozwoju.

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.