Politechnika Poznańska
Wydział Informatyki
Instytut Informatyki

Praca dyplomowa magisterska

INTEGRACJA DRZEW PREFIKSOWYCH W PRZETWARZANIU ZBIORÓW ZAPYTAŃ EKSPLORACYJNYCH ALGORYTMEM APRIORI

Szymon Dolata

Promotor
Dr inż. Marek Wojciechowski

Poznań, 2014 r.

Tutaj przychodzi karta pracy dyplomowej;
oryginał wstawiamy do wersji dla archiwum PP, w pozostałych kopiach wstawiamy ksero.

Spis treści

Rozdział 1

Wstęp

1.1 Integracja drzew prefiksowych w przetwarzaniu zbiorów zapytań eksploracyjnych algorytmem Apriori

Odkrywanie zbiorów częstych i generowanie na ich podstawie reguł asocjacyjnych, to problem sformułowany w kontekście analizy koszyka zakupów. Głównym celem jest szukanie prawidłowości w zachowaniu klientów supermarketów. Szybko znalazł on również zastosowanie w wielu innych dziedzinach, takich jak chociażby analiza działalności firm wysyłkowych, sklepów internetowych etc. Z wykorzystaniem znalezionych zbiorów częstych i wygenerowanych reguł dąży się do tego aby możn było wynioskować (z dużym prawdopodobieństwem), że niektóre produkty współwystępują ze sobą. Informacje takie, zwłaszcza jeśli wyrażone w formie zasad, często mogą być stosowane w celu zwiększenia sprzedanych danych produktów - na przykład poprzez odpowiednie rozmieszczenie ich na półkach w supermarkecie lub na stronach katalogu wysyłkowego (umieszczenie obok siebie może zachęcić jeszcze więcej klientów do zakupu ich razem) lub poprzez bezpośrednie sugerowanie klientom produktów, którymi mogą być zainteresowani.

Oczywistym jest, że należy szukać tylko takich reguł asocjacyjnych, które są wiarygodne i niosą ze sobą jakąś informację. Istnieją wskaźniki służące do oceny tychże reguł. Zostały one omówione bardziej szczegółowo w rozdziale 2.

Głównym problemem indukcji reguł asocjacyjnych jest to, że istnieje bardzo wiele możliwości. Przykładowo w zakresie produktów z supermarketu, których może być nawet kilka tysięcy, istnieją miliardy możliwych reguł. Tak ogromna ilość nie może być przetwarzana sekwencyjnie. Dlatego potrzebne są wydajne algorytmy, które ograniczają przestrzeń wyszukiwania i sprawdzają jedynie podzbiór wszystkich reguł. Jednym z takich algorytmów jest Apriori opracowany przez [**?**].

Podstawowy algorytm Apriori trzyma kandydatów w drzewie haszowym. W ostatnich latach zaproponowane zostały metody Common Counting ([**?**]) oraz Common Candidate Tree ([**?**]). Są one wynikiem badań nad optymalizacją wykonania kilku zadań Apriori uruchomionych współbieżnie na nakładających się podzbiorach tabeli z danymi. Metody sprowadzały się do:
- Integracji odczytów współdzielonych danych z dysku;
- Integracji drzew haszowych w jedno drzewo gdzie kandydaci mają kilka liczników (po jednym dla zadania eksploracji).
W praktyce jednak lepsze okazały się implementacje Apriori gdzie drzewo haszowe zastąpiono znacznie prostszą strukturą drzewa prefiksowego. Powstało kilka rozwiązań wykorzystujacych tę strukturę: Borgelt, Bodon, Goethals. Jednak do tej pory nie została zaimplementowana modyfikacja Common Counting i Common Candidate Tree dla Apriori z drzewem prefiksowym i to właśnie jest celem tej pracy.

1.2 Cel i zakres pracy

Tak jak wspomniano, ogolnym celem pracy jest implementacja dwóch algorytmów wykonania zbioru zapytań odkrywających zbiory częste - które dotychczas implementowne były na drzewie haszowym - z wykorzystaniem drzew prefiksowych.
Na ten ogólny cel pracy składają się następujące cele szczegółowe: - przedstawienie, analiza i porównanie istniejących rozwiązań dotyczących tematyki pracy - implementacja modyfikacji Common Counting i Common Candidate Tree dla Apriori z drzewem prefiksowym;

- przetesotowanie wydajności zaimplementowanych algorytmów.

1.3 Opis infrastruktury

Algorytmy napisane zostały w języku Java, z wykorzystaniem narzędzia Maven oraz środowiska programistycznego Eclipse. Dane testowe generowane były za pomocą generatora GEN ([**?**]), a następnie wczytywane do bazy PostgreSQL, z której korzystała aplikacja. Tetsy przeprowadzone zostały na komputerze HP Envy 14 Notebook PC, z procesorem Intel Core i5-2410M 2x2.30GHz oraz 8GB pamięci RAM, pracujacym pod kontrola systemu operacyjnego Microsoft Windows 7.

1.4 Struktura pracy

Struktura pracy jest następująca:
– w rozdziale 2 omówiono podstawowe pojęcia i definicje wykorzystywane w pracy;
– w rozdziale 3 przedstawiono istniejące rozwiązania i algorytmy, związane z tematem pracy;
– w rozdziale 4 przedstawiono ideę, opis i cechy alogrytmu;
– w rozdziale 5 przeanalizowano działanie algorytmu dla różnych parametrów i danych wejściowych;
– w rozdziale 6 przedstawiono winoski i uwagi do pracy.

Rozdział 2

Podstawowe pojęcia i definicje

2.1 Wstęp

W poniższym rozdziale omówiono podstawowe pojęcia i definicje wykorzystywane w pracy.

2.2 Lista pojęć i definicji

2.2.1 Transakcja i element transakcji

Danymi wejściowymi dla odkrywania zbiorów częstych i reguł asocjacyjnych jest zbiór transakcji zdefiniowanych na zbiorze elementów. Tymi elementami mogą być produkty w sklepie, usługi, książki etc. Ważne jest, aby te elementy można było w łatwy sposób od siebie odróżnić. Jeśli $I = \{i_1, i_2, \cdots, i_n\}$ (ang. *item base*), to zbiór wszystkich możliwych elementów, to dowolny niepusty podzbiór X zbioru $X \subseteq I$ nazywamy transakcją (ang. *itemset*). Natomiast zbiór elementów o mocy k, to taki zbiór, który posiada dokładnie k elementów (ang. *k-itemset*). Mówi się, że

Transakcja jest zatem przykładowym zbiorem elementów, np. zbiorem produktów, które zostały kupione przez danego klienta. Jako że transakcje mogą się powtarzać (może istnieć kilku klientów, którzy kupili dokładnie takie same produkty), to nie ma możliwości żeby zamodelować wszystkie możliwe transakcje (koszyki). Wynika to z tego, że elementy w zbiorze nie mogą się powtarzać. Problem ten znalazł kilka rozwiązań. Należy do nich zamodelowanie wszystkich transakcji jako multizbioru (uogólnienie pojęcia zbioru, w którym w odróżnieniu od klasycznych zbiorów jeden element może występować wiele razy) albo jako wektora (elementy na różnych pozycjach mogą być takie same, ale wyróżnia je położenie). Innym - choć podobnym do wspomnianego wyżej zastosowania wektora - rozwiązaniem jest rozszerzenie każdej transakcji o unikalny identyfikator. Kolejną możliwością jest wykorzystanie zbioru unikalnych transakcji, z tą różnicą, że do każdej transakcji przypisany jest licznik mający za zadanie zliczanie wystąpień.

Należy także zwrócić uwagę, że w większości rozważanych przypadków nie są znane wszystkie elementy, jakie mogą znaleźć się w zbiorze I. Przyjmuje się wówczas, że ten zbiór jest sumą elementów występujacych we wszystkich transakcjach.

2.2.2 Reguła asocjacyjna

Reguła asocjacyjna jest implikacją, która daje możliwość przewidywania jednoczesnego wystąpienia dwóch zjawisk i zachowań, współzależnych od siebie. Innymi słowy jest to schemat, pozwalający - z określonym prawdopodobieństwem - założyć, że jeśli nastąpiło zdarzenie A, to nastąpi również zdarzenie B. W kontekście problemu koszyka zakupów sprowadza się do reguł w stylu: *Jeżeli klient kupił pieluszki, to (z określonym prawdopodobieństem) kupi też piwo.*

2.2.3 Wsparcie transakcji

Jeśli T oznacza jedną z transakcji w zbiorze wszystkich transakcji D, to (bezwzgędne) wsparcie tej transakcji jest równe U - liczbie wystąpień T w zbiorze D. Wsparcie względne jest to z kolei procent (lub ułamek) transakcji w zbiorze D, które zawierają T. Obliczamy ze wzoru

$$sup_{rel}(T) = \frac{|U|}{|D|} * 100\%$$

. Dla algorytmu Apriori określa się próg minimalnego wsparcia *minsup*, który również może być wyrażony w dwojakiej postaci - jako liczba wystąpień lub procent wszystkich transakcji. W poszukiwaniu zbiorów częstych interesujące są tylko te reguły, dla których $sup(T) \geq minsup$, gdzie $sup(T)$, to przyjęty w pracy sposób zapisu wsparcia transakcji T w rozważanym zbiorze transakcji D.

2.2.4 Ufność reguły asocjacyjnej

Ufność reguły asocjacyjnej jest miarą jakości danej reguły. Miara ta została przedstawiona przez autorów algorytmu Apriori [**?**]. Dla reguły asocjacyjnej postaci $R = "X \rightarrow Y"$ (gdzie X i Y to zbiory elementów) ufność wyraża się jako stosunek wspracia sumy wsystkich elementów występujących w regule (w tym przypadku $sup(X \cup Y)$) do wstparcia poprzednika reguły (tutaj $sup(X)$).

$$conf(R) = \frac{sup(X \cup Y)}{sup(X)}$$

Należy dodać, że nie ma znaczenia czy wykorzystywane jest wsparcie aboslutne czy relatywne. Istotne jest natomiast to, aby w zarówno dla licznika i mianowika wykorzystany był ten sam typ wsparcia. Z powyższego wzoru wynika, że ufność reguły asocjacyjnej, to stosunek liczby przypadków, w których jest ona poprawna, do wszystkich przypdków gdzie mogłaby zostać zastosowana. Przykład: $R = wino \wedge chleb \rightarrow ser$ - jeśli klient kupuje wino i chleb, to ta reguła ma zastowanie i mówi, że można oczekiwać, że dany klient kupi również ser. Jest możliwe, że ta reguła - dla danego klienta - będzie poprawna lub nie. Interesują informacją jest to jak dobra jest reguła, czyli jak często jest poprwana (jak często klient, króty kupuje wino i chleb kupuje również ser). Taką właśnie informację uzyskuje się poprzez obliczenie ufności reguły asocacyjnej. Oczywiście w przypadku gdy klient nie kupił chleba lub/i wina, to reguła nie znajduje zastowania, a dana transakcja nie wpływa na $conf(R)$.

2.2.5 Wsparcie reguły asocjacyjnej

Wsparcie reguły asocjacyjnej postaci $A \cup B \rightarrow C$ odpowiada wsparciu zbioru $S = \{A, B, C\}$ ([**?**]).Miara ta informuje o tym jak często dana reguła jest prawidłowa. Nieco odmienna definicja została przedstawiona i wykorzystana w [**?**]. Różnica polega na tym, że wpsarcie wyrażone jest jako liczba przypdków, w których reguła jest stosowalna. Zatem dla powyżej postaci byłoby to $S = \{A, B\}$, nawet jeśli reguła może okazać się fałszywa. Wparcie może być stosowane do filtrowania. Dla ustalonego *minsup* szuka się tylko takich reguł, których wsparcie jest nie mniejsze od *minsup*. Oznacza to, że interesujące są tylko te reguły, które wystąpiły co najmniej daną liczbę razy. W algorytmach wyszukiwania reguł asocjacyjnych stosuje się progi mnimalnego wsparcia oraz mnimalnej ufności. Dzięki temu w otrzymanych wynikach nie są uwzględnione mało wartościowe reguły.

2.2.6 Zbiór częsty

Zbiorem częstym nazywamy taki niepusty podzbiór zbioru I, dla którego wsparcie jest równe co najmniej wartości *minsup*.

Rozdział 3

Podłoże teoretyczne

3.1 Wstęp

Kolejny rozdział przedstawia aktualne metody i istniejące algorytmy związane z tematem pracy. Poza podstawowym algorytmem Apriori ([**?**]), który używa drzew haszowych do przechwywania kandydatów, opisano trzy modyfikacje tego alogrytmu. Główna różnica polega na tym, że wykorzystują one inną strukturę, a mianowicie drzewa prefiksowe. Są to rozwiązania zaproponowane przez Christina Borgelta ([**?**]), Ferenca Bodona ([**?**]) oraz Barta Goethalsa ([**?**]). Ze względu na wykorzystanie prostszej struktury okazały się one szybsze od standardowego algorytmu.

Innym problemem jest optymalizacja wykonania kilku zadań Apriori uruchomionych współbieżnie na nakładających się podzbiorach tabeli z danymi. Metody z tym związane to Common Counting ([**?**]) i Common Candidate Tree ([**?**]). Oparte są one o implementację Apriori z zastosowaniem drzew haszowych. Brakuje jednak adaptacji tych algorytmów, polegającej na zmianie struktury na drzewa prefiksowe. Właśnie taka modyfikacja została wprowadzona, a uzyskane efekty opisano w kolejnych rozdziałach niniejszej pracy.

3.2 Przegląd istniejących rozwiązań

3.2.1 Algorytm Apriori [?]

Algorytm Apriori jest algorytmem eksploracji poziomej. Szuka zbiorów częstych o rozmiarach $1, 2, \ldots, k$. Algorytm rozpoczyna od zbiorów o rozmiarze 1 i następnie zwiększa ten rozmiar w kolejnych iteracjach. Elementy każdej transakcji są uporządkowane leksykograficznie - jeżeli nawet transakcje nie są posortowane, to krokiem wstępnym algorytmu może być leksykograficzne uporządkowanie elementów transakcji ([**?**]). Po pierwszym kroku zebrane są zatem wszystkie elementy występujące w transakcjach (w postaci zbiorów jednolementowych). Następnie sprawdzane jest, które z nich posiadają wsparcie nie mniejsze niż *minsup*. Elementy niespełniające tego wymgania są odrzucane. Pozostałe służą do utworzenia dwuelementowych zbiorów kandydujących (ang. *candidate itemsets*). Dla wygenerowanych zbiorów spradzane jest czy posiadają wsparcie równe co najmniej *minsup*. Jeśli tak, to taki zbiór jest dodawany do listy zbiorów częstych i w kolejnej iteracji jest wykorzystywany (wraz z innymi zbiorami z tejże listy) do generowania zbiorów kandydatów o rozmiarze o 1 większym. Wsparcie zbiorów sprawdzane jest na podstawie odczytu danych z bazy danych. Algorytm zatrzymuje się gdy nie ma już możliwości generowania kolejnych zbiorów. W wyniku jego działania zwracana jest suma k-elementowych zbiorów częstych ($k = 1, 2, \ldots$), która może zostać wykozystana do generowania reguł asocjacyjnych.

3.2.2 Algorytm Apriori - implementacja Christina Borgelta [?]

3.2.3 Algorytm Apriori - implementacja Ferenca Bodona [?]

3.2.4 Algorytm Apriori - implementacja Barta Goethalsa [?]

3.2.5 Common Counting [?]

W metodzie Common Counting chodzi o równoległe wykonanie zbioru zapytań eksploracyjnych algorytmem Apriori z integracją porywających się fragmentów bazy danych. Na wejściu algorytm otrzymuje zbiór elementarnych predykatów selekcji danych dla zbioru zapytań eksploracyjnych

DMQ. Początkowo algorytm ustala zbiór wszystkich elementów, czyli takich, które wystąpiły w co najmniej jednej transakcji. W kolejnych krokach generowane są zbiory częste odzielnnie dla każdego z zapytań. Przebiega to w taki sam sposób jak w przypadku standardowego alogrytmu Apriori. Z każdym zapytaniem powiązane jest drzewo haszowe, w którym przechowywani są kandydaci. Warunek zatrzymania algorytmu jest taki jak w standardowym Apriori (brak możliwości wygenerowania kandyadtów w kolejnej iteracji), z tą różnicą, że musi być spełniony dla wszystkich zapytań ze zbioru. Zliczenie wystąpień kandyadatów jest realizowane dla wszystkich zapytań jednocześnie. Partycje bazy danych są odczytywane sekwencyjnie dla poszczególnych elementarnych predykatów selekcji danych. Powiększeniu ulegają liczniki kandydatów zawartych w analizowanej transakcji dla zapytań posiadających odwołania do danej partycji. Lista kandydatów zawiarających się w danej transakcji ustalana jest poprzez testowanie transakcji względem drzew haszowych. Należy tutaj zaznaczyć, że w przypadku gdzy kilka zapytań współdzieli dany elementarny predykat selekcji danych, to podczas zliczeń wystąpień kandydatów odczyt właściwej mu partycji jest wykonywany tylko raz. Zatem optymalizowane są odczyty współdzielonych przez zapytania fragmentów bazy danych, przy czym pozostałe kroki algorytmu Apriori pozostają niezmienione i są wykonywane oddzielnie dla każdego zapytania.

3.2.6 Common Candidate Tree [?]

Common Candidate Tree podobna do Common Counting. Również korzysta z oryginalnego Apriori i wykorzystuje strukturę drzewa haszowego. Różnica polega na tym, że zwiększony został stopień współbieżności przetwarzania. Uzyskano to dzięki współdzieleniu pamięciowej struktury drzewa składującego kandydatów. Jest to duża zaleta w porównaniu z Common Counting, gdyż - zamiast wielu - tworzone jest jedno drzewo haszowe o niezmiennej strukturze. Poza zachowaniem integracji odczytów współdzielonych możliwa jest integracja testowania czy w danej transakcji zawierają się kandydaci z poszczególnych zapytań. Realizacja tego alogrytmu wymagała rozszerzenia struktury kandydatów. W jej wyniku z każdym kandydatem związany został wektor liczników (jeden licznik dla jednego zapytania), a nie pojedyńczy licznik. Dodatkowo - dla rozróżnienia zapytań, które wygenerowały danego kandydata - dołączony został wektor flag logicznych przechowujący taką właśnie informację. Po wyłonieniu kadydatów są oni umieszczani w jednym zbiorze. Zbiór ten trafia do wspólnego drzewa haszowego. W tym kroku modyfikowane są również odpowiednie falgi. Samo generowanie kandydatów i selekcja zbiorów czestych nadal realizowane sa odrebnie dla poszczególnych zapytań. Zliczany jest natomiast zintegrowany zbiór kandydatów. Podczas tej fazy brani są pod uwagę tylko kandydaci wygenerowani przez zapytania odwołujące się do aktualnie odczytywanej partycji bazy danych i w przypadku gdy kandydat zawiera się w przetwarzanej transakcji, to zwiększa się liczniki kandydatów związne z tymi zapytaniami. Eksperymenty [**?**] pokazały, że jest to alogrytm wydajniejszy i lepiej skalowany od Common Counting.

Rozdział 4

Opracowane algorytmy

4.1 Wstęp

Whatever

4.2 Common Steps

Whatever

Rozdział 5

Wyniki eksperymentów

5.1 S1

5.2 S2

Rozdział 6

Wnioski i uwagi

Whole conclusion for one page.

Dodatek A

Content of the DVD

As an addition to this document, the DVD is attached. It provides some materials connected with the presented subject in electronic form for potential users or people, who would want to continue works on this topic.

The DVD content consists of several items:

1. Item 1
2. Item 2
3. Item 3

Instytut Informatyki, Wydział Informatyki
Politechnika Poznańska

Skład przy użyciu systemu LaTeX.

BibTeX:

```
@mastersthesis{ key,
    author = "Szymon Dolata",
    title = "{Integracja drzew prefiksowych w przetwarzaniu zbiorów zapytań eksploracyjnych
algorytmem Apriori}",
    school = "Poznan University of Technology",
    address = "Pozna{\'n}, Poland",
    year = "2014",
}
```