algorithm เพื่อแปลงข้อมูลและวัดความหนาแน่นของข้อมูลในแต่ละ window ข้อมูลที่ถูกแปลงแล้วแต่ยังไม่มีการสุ่มเพื่อลดขนาดข้อมูล จะถูกส่งไปยังโปรแกรม k-means เพื่อทดสอบผลการจัดกลุ่ม ในขณะเดียวกันข้อมูลที่ถูกแปลงและวัดความหนาแน่นแล้วจะถูกส่งต่อไปยังโมดูล Sampling techniques เพื่อทดสอบวิธีการสุ่มในแบบต่างๆ ซึ่งประกอบด้วย

- Simple random sampling เป็นวิธีการสุ่มเลือกข้อมูลอย่างง่าย โดยไม่มีการพิจารณาความหนาแน่นของข้อมูล
- Density-biased sampling เป็นวิธีการสุ่มเลือกข้อมูล โดยคัดเลือกจากข้อมูลที่มีความหนาแน่นถึงเกณฑ์ที่กำหนด โดยมีการใช้โครงสร้างข้อมูลประกอบกับเทคนิคการคัดเลือกข้อมูลดังนี้
  - Hashing ใช้เนื้อที่เก็บข้อมูลขนาดใหญ่ (ด้วยสมมติฐานว่าข้อมูลที่ความหนาแน่นถึงเกณฑ์ทั้งหมด สามารถเก็บลงตารางแฮชได้) และมีวิธีการเข้าถึงข้อมูลโดยตรง
  - Reservoir + Hashing ใช้เนื้อที่เก็บข้อมูลที่มีขนาดจำกัด (ด้วยโครงสร้างข้อมูลแบบแถวลำดับที่เรียกว่า reservoir) และมีวิธีการเข้าถึงข้อมูลได้โดยตรง (วิธีนี้เป็นเทคนิคแบบเดียวกับที่ Palmer และ Faloutsos (2000) ใช้ในงานวิจัย)
  - Reservoir + Simple random sampling (SRS) ใช้เนื้อที่เก็บข้อมูลที่มีขนาดจำกัด และใช้วิธีการคัดเลือกข้อมูลเพื่อเก็บลง reservoir แบบการสุ่มอย่างง่าย
  - Reservoir + Rejection sampling ใช้เนื้อที่เก็บข้อมูลที่มีขนาดจำกัด และใช้การสุ่มสองครั้ง (rejection sampling) เพื่อพิจารณาว่าจะทิ้งข้อมูล หรือจะเก็บข้อมูลลงเนื้อที่ reservoir

ข้อมูลที่ได้รับการสุ่มเลือกด้วยเทคนิคต่างๆกัน จะถูกนำไปจัดกลุ่มด้วยโปรแกรม k-means เพื่อพิจารณาค่า cluster means ที่เป็นผลลัพธ์ของการจัดกลุ่ม เปรียบเทียบกับค่า real centroids เทคนิคการสุ่มเลือกที่ดีควรจะต้องให้ผลลัพธ์การจัดกลุ่มที่เบี่ยงเบนไปจาก real centroids น้อยที่สุด ชุดคำสั่งในภาษาเออแลงส่วนที่ทำหน้าที่สุ่มข้อมูลตามความหนาแน่นแสดงได้ดังรูปที่ 2 และโปรแกรมสุ่มข้อมูลด้วยอัลกอริทึม k-means แสดงได้ดังรูปที่ 3

```
%=== Density-biased sampling
%      Firstly, generate back from slidingWindows -> POINTs
genWin2P([])->[];          % input arguments: WindowsList,Den
genWin2P([{P,Den}|T])-> if Den>=1 -> dup(center(P),Den)++genWin2P(T);
                           true -> genWin2P(T)
                        end.
%---   Then, select sampling technique as: Hash-based sampling
specificDensBin(Bin,L,Den) -> take(Bin,specificDens(L,Den)).
specificDens([],_)->[];
specificDens([{P,D}|T],Den)-> if D>=Den -> [{P,D}|specificDens(T,Den)];
                                 true -> specificDens(T,Den)
                              end.
%---   Or: Simple Random with density bias
simpleRandom(_,_,0)->[];
simpleRandom(WindowL,Dens,Bin)->
        Nth=random:uniform(length(WindowL)),
        {P,D}=lists:nth(Nth,WindowL),
        if D>=Dens -> [{P,D}|simpleRandom(WindowL,Dens,Bin-1)]  ;
              true-> simpleRandom(WindowL,Dens,Bin)
        end.
%---  Or: Rejection sampling with density bias
rejectionRandom(_,_,_,0)->[];
rejectionRandom(Para,WindowL,Dens,Bin)->
        Nth=random:uniform(length(WindowL)),
        Par=random:uniform(),
        {P,D}=lists:nth(Nth,WindowL),
        if (0.5-Para)=<Par, Par=<(0.5+Para),D>=Dens ->
                     {{P,D}|rejectionRandom(Para,WindowL,Dens,Bin-1)]  ;
              true-> rejectionRandom(Para,WindowL,Dens,Bin)
        end.
% ---------------------
```

รูปที่ 2. ชุดคำสั่งภาษาเออแลงที่ทำหน้าที่สุ่มข้อมูลตามความหนาแน่น

```
kMeans(PL)-> {_,N}=read('enter number of clustering>'),
             CL=take(N,sets:to_list(sets:from_list(PL))), %initial centroid
             clustering(1,CL,PL).

take(0,_)->[]; % take firts  distinct-n element of list
take(N,[H|T])->[H|take(N-1,T)].
clustering(N,CL,PL)->
          L1= lists:map( fun(A) -> nearCentroid(A,CL) end ,PL),
          L2 = transform(CL,L1),
          NewCentroid = lists:map(fun({_,GL})-> findMeans(GL)end, L2),
          N1= N+1,
          if NewCentroid==CL -> NewCentroid; % return new centroid
                       N>=90 -> NewCentroid;  % max iterations=90
                       true ->  clustering(N1,NewCentroid,PL)
          end.
nearCentroid(Point,CentroidL)->
    LenList=lists:zip(lists:map(fun(A)-> distance(Point,A)end,CentroidL),CentroidL),
    [{_,Centroid}|_]=lists:keysort(1,LenList),
    {Point,Centroid}.

% transform Point-CentroidList to Centroid-PointList
transform([],_)->[];
transform([C|TC],PC)->[{C,t1(C,PC)} |transform(TC,PC)].
t1(_,[])->[] ;
t1(C1,[H|T])->{P,C}=H,
              if C1==C -> [P|t1(C1,T)];
                 C1=/=C -> t1(C1,T)
              end.
% compute Euclidean distance
distance([],[])-> 0;
distance([X1|T1],[X2|T2])-> math:sqrt((X2-X1)*(X2-X1)+distance(T1,T2) ).

% findMeans([[1,2],[3,4]]) --> [2.0,3.0]
findMeans(PointL) ->[H|_]=PointL ,Len=length(H),
      AllDim=lists:reverse(allDim(Len,PointL)),
      lists:map(fun(A)-> mymean(A) end, AllDim ).
mymean(L)->lists:sum(L)/length(L).

allDim(0,_)->[];
allDim(D,L)->[eachDimList(D,L)|allDim(D-1,L)].
eachDimList(_,[]) ->[];
eachDimList(N,[H|T]) ->[lists:nth(N,H)|eachDimList(N,T)].
% ------------------------
```

รูปที่ 3. โปรแกรม k-means clustering ในรูปแบบของภาษาเออแลง

ผลการทดลองและวิจารณ์

การทดสอบโปรแกรมของโครงการวิจัยนี้ มีวัตถุประสงค์หลักเพื่อการทดสอบประสิทธิภาพของโปรแกรมสุ่มข้อมูลตามความหนาแน่นทั้ง 4 เทคนิคได้แก่ (1) Density-biased sampling: hashing (unlimited number of bins), (2) Density-biased sampling: reservoir and hashing, (3) Density-biased sampling: reservoir and simple random sampling, และ (4) Density-biased sampling: reservoir and rejection sampling และเพื่อความสมบูรณ์ในการเปรียบเทียบ ผู้วิจัยได้เพิ่มเติมการจัดกลุ่มข้อมูลกับข้อมูลที่ถูกสุ่มอย่างง่าย โดยไม่มีการพิจารณาค่าความหนาแน่นและไม่ต้องมีการแปลงข้อมูลด้วยเทคนิค window sliding โดยกำหนดขนาดและจำนวนของ bin หรือเนื้อที่เก็บข้อมูลในโครงสร้าง reservoir ให้เท่ากับที่ใช้ในเทคนิคที่ (2), (3) และ (4) ส่วนในเทคนิคที่ (1) จะใช้จำนวน bin ให้มากที่สุดเท่าที่เนื้อที่หน่วยความจำขณะนั้นจะจัดสรรให้ได้ ทั้งนี้เพื่อใช้เป็นเกณฑ์เปรียบเทียบประสิทธิภาพการใช้เนื้อที่หน่วยความจำกับเทคนิคที่ (2)-(4)

การจัดกลุ่มตามความหนาแน่น มีวัตถุประสงค์ที่จะพัฒนาเทคนิคการจัดกลุ่มกับข้อมูลขนาดใหญ่ที่สามารถพัฒนาต่อไปให้จัดการกับข้อมูลสตรีมได้ ดังนั้นเทคนิคการสุ่มและการจัดกลุ่มที่ดี จะต้องใช้เนื้อที่หน่วยความจำต่ำ แต่ให้ผลการจัดกลุ่มที่คลาดเคลื่อนไปจากกลุ่มที่แท้จริงเป็นระยะทางน้อยที่สุด ดังนั้นการสรุปผลเทคนิคต่างๆที่ใช้ในการจัดกลุ่ม กำหนดเกณฑ์ในการเปรียบเทียบว่าแต่ละเทคนิคจะต้องใช้หน่วยความจำไม่มากกว่า 30% ของหน่วยความจำที่โปรแกรม k-means ใช้ในการ

จัดกลุ่มข้อมูลสังเคราะห์ทั้งหมด เช่น ถ้าการรัน k-means กับข้อมูลทั้งหมดใช้หน่วยความจำ 4,800 ไบต์ เทคนิคการสุ่มและการจัดกลุ่มจะต้องใช้หน่วยความจำไม่เกิน 1,440 ไบต์ ผลการเปรียบเทียบความถูกต้องของการจัดกลุ่มข้อมูลแสดงได้ดังตารางที่ 1 โดยค่าความคลาดเคลื่อนที่แสดงในตารางเป็นค่าเฉลี่ยจากการทดสอบในทุกค่า density และทุกค่า bin เครื่องหมาย '--' ที่ปรากฏในตารางหมายถึงเทคนิคนั้นๆใช้หน่วยความจำมากกว่าเกณฑ์ขั้นต่ำที่กำหนด จึงไม่สามารถนำผลการจัดกลุ่มมาเปรียบเทียบร่วมกับเทคนิคอื่นๆได้

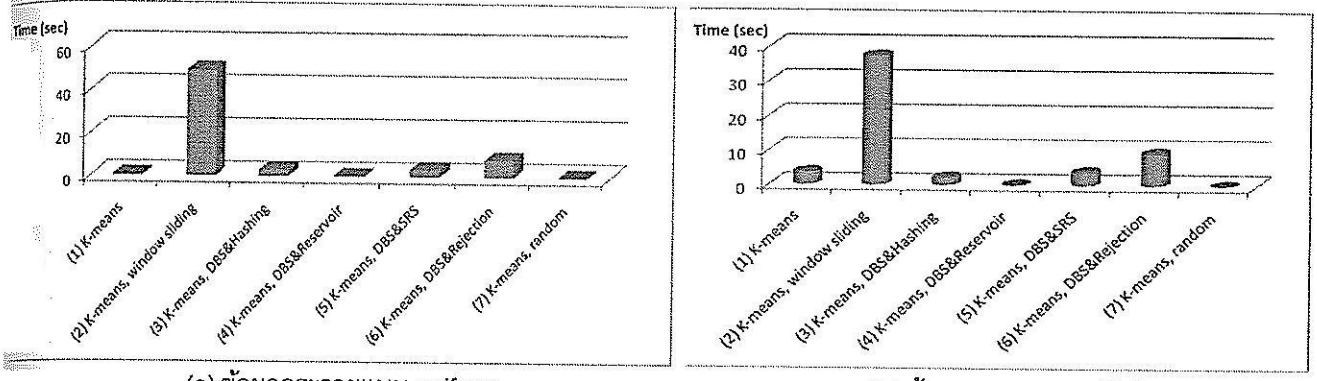
ตารางที่ 1. การเปรียบเทียบค่าความคลาดเคลื่อนของกลุ่มข้อมูลเมื่อใช้เทคนิคการสุ่มแต่ละแบบ

ลักษณะของข้อมูล		K-means, DBS &Hashing	K-means, DBS &Reservoir	K-means, DBS&SRS	K-means, DBS &Rejection	K-means, random
ข้อมูลกระจายสม่ำเสมอ						
ขนาดเล็ก	2 มิติ, 4 กลุ่ม	--	--	6.067	6.029	4.552
	2 มิติ, 8 กลุ่ม	--	10.739	14.661	16.471	13.930
	3 มิติ, 4 กลุ่ม	--	8.032	9.479	10.723	8.709
	3 มิติ, 8 กลุ่ม	--	55.889	26.051	25.071	31.979
ขนาดใหญ่	2 มิติ, 4 กลุ่ม	--	3.557	6.600	6.800	13.520
	2 มิติ, 8 กลุ่ม	--	15.22	13.721	15.385	15.446
	3 มิติ, 4 กลุ่ม	--	--	--	--	44.721
	3 มิติ, 8 กลุ่ม	--	48.965	29.364	31.560	20.318
ข้อมูลกระจายแบบซิฟ						
ขนาดเล็ก	2 มิติ, 4 กลุ่ม	--	20.377	15.098	14.124	14.634
	2 มิติ, 8 กลุ่ม	--	27.889	23.689	23.350	24.704
	3 มิติ, 4 กลุ่ม	--	45.873	16.320	17.525	17.589
	3 มิติ, 8 กลุ่ม	--	90.360	33.534	34.580	34.083
ขนาดใหญ่	2 มิติ, 4 กลุ่ม	--	37.888	18.310	15.228	19.528
	2 มิติ, 8 กลุ่ม	--	32.780	31.408	30.598	35.520
	3 มิติ, 4 กลุ่ม	--	30.481	11.709	13.003	14.503
	3 มิติ, 8 กลุ่ม	--	39.551	27.707	28.434	32.479

เมื่อพิจารณาจากผลการทดลองตามตารางที่ 1 จะเห็นได้ว่าเทคนิค K-means, DBS& Hashing เมื่อทดสอบกับข้อมูลทุกชุด ปรากฏผลเป็นเครื่องหมาย '--' ทั้งนี้เนื่องจากเทคนิคนี้ไม่มีข้อกำหนดเกี่ยวกับจำนวน bin ทำให้ใช้หน่วยความจำมากกว่าเกณฑ์ที่ตั้งไว้ จากผลการเปรียบเทียบค่าความคลาดเคลื่อนของกลุ่มข้อมูล จะเห็นได้ว่าเมื่อกลุ่มข้อมูลมีการกระจายแบบสม่ำเสมอและเป็นข้อมูลขนาดเล็ก เทคนิค K-means, DBS&Reservoir ที่ใช้การเข้าถึงข้อมูลแบบแฮชจะให้ผลลัพธ์ที่ดี แต่ถ้าข้อมูลมีขนาดใหญ่เทคนิค K-means, random จะให้ผลลัพธ์การจัดกลุ่มที่ดี ในกรณีที่ข้อมูลกระจายแบบซิฟและเป็นข้อมูลขนาดสองมิติ เทคนิค K-means, DBS& Rejection จะให้ผลลัพธ์การจัดกลุ่มที่ดี แต่เมื่อข้อมูลเพิ่มขนาดเป็นสามมิติ เทคนิค K-means, DBS& SRS จะให้ผลลัพธ์การจัดกลุ่มที่ดีกว่า

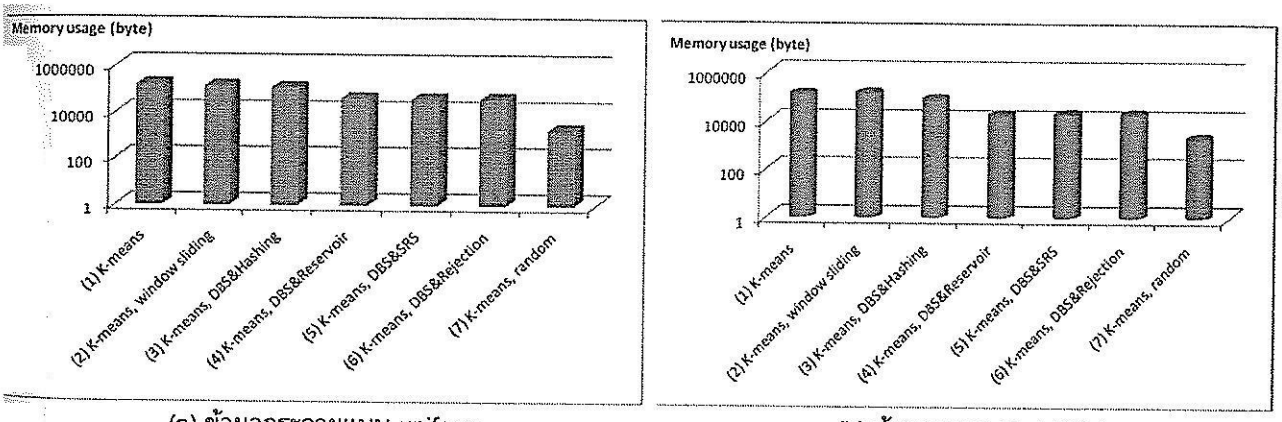
การเปรียบเทียบเวลาในการประมวลผล จะรวมเวลาทั้งในขั้นตอนการสุ่มและขั้นตอนการจัดกลุ่มของทั้ง 7 เทคนิค ประกอบด้วย (1) K-means, (2) K-means, window sliding, (3) K-means, DBS&Hashing, (4) K-means, DBS&Reservoir, (5) K-means, DBS&SRS, (6) K-means, DBS&Rejection และ (7) K-means, random

การแสดงผลทดสอบเปรียบเทียบเวลาและการใช้เนื้อที่หน่วยความจำในรูปที่ 4 และ 5 ตามลำดับ จะแสดงเฉพาะกรณีของข้อมูล 3 มิติ เป็นข้อมูลขนาดใหญ่จำนวน 8 กลุ่ม (ข้อมูลลักษณะอื่นๆ ให้ผลสอดคล้องเช่นเดียวกัน) โดยแสดงเปรียบเทียบทั้งกรณีข้อมูลกระจายสม่ำเสมอ และข้อมูลกระจายแบบซิฟ



(a) ข้อมูลกระจายแบบ uniform (b) ข้อมูลกระจายแบบ Zipf

รูปที่ 4. เปรียบเทียบเวลาที่ใช้ในการจัดกลุ่มข้อมูล 3 มิติ ขนาดใหญ่และมีจำนวน 8 กลุ่ม



(a) ข้อมูลกระจายแบบ uniform (b) ข้อมูลกระจายแบบ Zipf

รูปที่ 5. เปรียบเทียบหน่วยความจำที่ใช้ในการจัดกลุ่มข้อมูล 3 มิติ ขนาดใหญ่และมีจำนวน 8 กลุ่ม

บทสรุป

งานวิจัยนี้ศึกษาวิธีการลดขนาดของข้อมูลด้วยเทคนิคการสุ่ม โดยคำนึงถึงลักษณะการกระจายของข้อมูล ถ้าข้อมูลมีการกระจายอย่างสม่ำเสมอการสุ่มข้อมูลจะให้น้ำหนักข้อมูลทุกตัวเท่าเทียมกัน แต่ถ้าข้อมูลมีการกระจายไม่สม่ำเสมอเช่นมีการกระจายแบบซิฟ ข้อมูลในบางช่วงมีความหนาแน่นมากกว่าข้อมูลในช่วงอื่นๆด้วยสัดส่วนความหนาแน่นที่แตกต่างกันมาก การสุ่มข้อมูลจะเพิ่มโอกาสการถูกสุ่มให้กับข้อมูลในช่วงที่มีความหนาแน่นสูง และเรียกเทคนิคการสุ่มแบบนี้ว่า การสุ่มตามความหนาแน่น (density-biased sampling, DBS) โดยการวัดความหนาแน่นของข้อมูลจะใช้เทคนิคการเลื่อนกรอบหน้าต่าง (window sliding) ซึ่งเป็นกรอบขนาดเล็กที่เคลื่อนไปบนแกนข้อมูลและวัดจำนวนจุดข้อมูลที่ปรากฏภายในแต่ละกรอบ เพื่อบันทึกเป็นค่าความหนาแน่นของกรอบหน้าต่าง การสุ่มตามความหนาแน่นจึงเป็นการคัดเลือกกรอบหน้าต่างที่มีจำนวนจุดภายในกรอบตรงตามเกณฑ์ที่กำหนด จากนั้นใช้พิกัดของกรอบหน้าต่างเหล่านี้สร้างกลับเป็นจุดข้อมูลที่มีจำนวนจุดเท่ากับค่าความหนาแน่นของกรอบหน้าต่าง

งานวิจัยนี้ได้ทดสอบการจัดกลุ่มข้อมูลกับข้อมูลที่ผ่านกระบวนการสุ่ม ทั้งที่เป็นการสุ่มอย่างง่ายที่ไม่มีการพิจารณาความหนาแน่นของข้อมูล และการสุ่มตามความหนาแน่นของข้อมูล ในการทดสอบได้ใช้ข้อมูลสังเคราะห์รวมทั้งหมด 16 ชุด ข้อมูลแปดชุดแรกเป็นข้อมูลสองมิติ และข้อมูลอีกแปดชุดเป็นข้อมูลสามมิติ ในข้อมูลแปดชุดแบ่งย่อยออกเป็นข้อมูลที่มีการกระจายอย่างสม่ำเสมอ จำนวนสี่ชุดที่มีทั้งข้อมูลขนาดเล็กและข้อมูลขนาดใหญ่ และข้อมูลที่มีการกระจายแบบซิฟ จำนวนสี่ชุดที่มีทั้งขนาดเล็กและใหญ่เช่นเดียวกัน เทคนิคการสุ่มข้อมูลประกอบด้วย (1) DBS: hashing (unlimited number of bins), (2) DBS: reservoir and hashing, (3) DBS: reservoir with simple random sampling, (4) DBS: reservoir with rejection sampling, และ (5) Simple random sampling on the original data set (no window sliding technique)

ผลการทดสอบพบว่าในข้อมูลที่มีการกระจายสม่ำเสมอ เทคนิคที่ 2 และ 5 ให้ผลการจัดกลุ่มที่ค่อนข้างดี กลุ่มข้อมูลที่ได้คลาดเคลื่อนไปจากกลุ่มที่แท้จริงเพียงเล็กน้อย แต่มีข้อสังเกตว่าเทคนิคที่ 5 ที่เป็นการสุ่มเลือกข้อมูลแล้วนำข้อมูลนั้นไปจัดกลุ่มด้วยโปรแกรม k-means จะใช้เนื้อที่หน่วยความจำน้อยมากเมื่อเทียบกับเทคนิคอื่นๆ ดังนั้นจึงอาจสรุปได้ว่าเมื่อข้อมูลมีการกระจายสม่ำเสมอ ไม่จำเป็นต้องทำการสุ่มตามความหนาแน่น เพราะเทคนิคนี้จะใช้เวลาและหน่วยความจำมากกว่าการสุ่มข้อมูลแบบปกติ และผลการจัดกลุ่มที่ได้จะไม่ดีขึ้นมากอย่างมีนัยสำคัญ แต่ในกรณีที่ข้อมูลมีการกระจายแบบซิฟ นั่นคือข้อมูลในแต่ละกลุ่มมีขนาดเล็ก-ใหญ่ต่างกันมาก และปริมาณข้อมูลในกลุ่มหรือความหนาแน่นก็แตกต่างกันมากเช่นเดียวกัน กรณีเช่นนี้พบว่าการสุ่มตามความหนาแน่นจะมีผลช่วยให้ข้อมูลที่คัดเลือกมาเพื่อการจัดกลุ่มได้ข้อมูลตัวแทนที่ดี ความหนาแน่นของข้อมูลที่ถูกคัดเลือกใกล้เคียงกับความหนาแน่นที่ปรากฏในกลุ่มข้อมูลดั้งเดิมที่สังเคราะห์ขึ้น

เทคนิคการสุ่มที่ทำงานได้ดีกับข้อมูลที่กระจายแบบซิฟ คือเทคนิคที่ 3 และ 4 นั่นคือเทคนิคที่มีการวัดความหนาแน่นข้อมูลด้วยการทำ window sliding คัดเลือกข้อมูลด้วยเกณฑ์ความหนาแน่นขั้นต่ำ จากนั้นนำกรอบหน้าต่างที่มีความหนาแน่นถึงเกณฑ์แปลงข้อมูลกลับ ให้ได้จำนวนจุดข้อมูลเท่ากับค่าความหนาแน่นของกรอบหน้าต่าง ขั้นตอนต่อจากนั้นเป็นการสุ่มเลือกข้อมูลให้มีปริมาณเท่ากับเนื้อที่ reservoir โดยการสุ่มในขั้นนี้อาจจะเป็นการสุ่มแบบ simple random sampling หรืออาจจะใช้วิธี rejection sampling ก็ได้ ซึ่งจากผลการทดลองพบว่าให้ผลลัพธ์ที่ใกล้เคียงกัน แต่วิธี simple random sampling จะใช้เนื้อที่หน่วยความจำ และใช้เวลาที่น้อยกว่าเล็กน้อย

กิตติกรรมประกาศ

งานวิจัยนี้ดำเนินการโดยหน่วยวิจัยด้านวิศวกรรมข้อมูลและการค้นหาความรู้ และได้รับการสนับสนุนงบประมาณจากมหาวิทยาลัยเทคโนโลยีสุรนารี สำนักงานกองทุนสนับสนุนการวิจัยและสำนักงานคณะกรรมการวิจัยแห่งชาติ

เอกสารอ้างอิง

[1] S. Guha, R. Rastogi, & K. Shim, Cure: An efficient clustering algorithm for large databases, *Proceedings of the ACM SIGMOD Int. Conf. on Management of Data*, 1998, 73-84.

[2] A.K. Jain & R.C. Dubes, *Algorithms for Clustering Data* (Englewood Cliffs, New Jersey: Prentice Hall, 1988).

[3] G. Karypis, E.-H. Han, & V. Kumar, Chameleon: A hierarchical clustering algorithm using dynamic modeling, *IEEE Computer*, 32(8), 1999, 68-75.

[4] G. Kollios, D. Gunopulos, N. Koudas, & S. Berchtold, Efficient biased sampling for approximate clustering and outlier detection in data sets, *IEEE Transactions on Knowledge and Data Engineering*, 15(5), 2003, 1170-1187.

[5] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probabilisty*, Vol.1, 1967, 281-297.

[6] C. Palmer & C. Faloutsos, Density biased sampling: An improve method for data mining and clustering, *Proceedings of the ACM SIGMOD Int. Conf. on Management of Data*, 2000, 82-92.

[7] J. Sander, M. Ester, H.-P. Kriegel, & X. Xu, Density-based clustering in spatial databases: The algorithm gdbscan and its application, *Data Mining and Knowledge Discovery*, 2(2), 1998, 169-194.

[8] A. Stehl, J. Ghosh, & R. Mooney, Impact of similarity measures on web-page clustering, *Proceedings of the Workshop of Artificial Intelligence for Web Search, 17th National Conference on Artificial Intelligence*, 2000.

[9] J.S. Vitter, Random sampling with a reservoir, *ACM Transaction on Mathematical Software*, 11(1), 1985, 37-57.

การประมวลผลหลังกระบวนการทำเหมืองข้อมูล

Post-Data Mining Processing

นิตยา เกิดประสพ และ กิตติศักดิ์ เกิดประสพ
หน่วยวิจัยวิศวกรรมข้อมูลและการค้นหาความรู้ สาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จ.นครราชสีมา
E-mail: nittaya@sut.ac.th
Nittaya Kerdprasop and Kittisak Kerdprasop
Data Engineering and Knowledge Discovery (DEKD) Research Unit, School of Computer Engineering,
Suranaree University of Technology, Nakhon Ratchasima 30000

บทคัดย่อ

กระบวนการทำเหมืองข้อมูลประกอบด้วยสามขั้นตอนหลัก คือก่อนการทำเหมืองข้อมูล ขณะทำเหมืองข้อมูล และหลังการทำเหมืองข้อมูล ระบบทำเหมืองข้อมูลส่วนมากจะสิ้นสุดกระบวนการที่การนำเสนอความรู้ในขั้นตอนขณะทำเหมืองข้อมูล แต่ในงานวิจัยนี้ได้นำเสนอขั้นตอนต่อจากนั้น คือ ขั้นตอนการใช้ประโยชน์จากความรู้ที่ได้จากการทำเหมืองข้อมูล งานวิจัยนี้แสดงวิธีการนำความรู้ในรูปแบบของกฎการจำแนกมาใช้ประโยชน์ โดยความรู้ในรูปแบบของกฎการจำแนก จะถูกนำมาประเมินคุณภาพด้วยเกณฑ์ความครอบคลุมข้อมูล กฎที่ครอบคลุมข้อมูลได้มาก จะได้รับการคัดเลือกมาใช้ในระบบผู้เชี่ยวชาญ การทดสอบความสามารถของระบบผู้เชี่ยวชาญที่ใช้การสร้างฐานความรู้แบบอัตโนมัตินี้ ใช้วิธีวัดความถูกต้องของคำแนะนำที่ให้โดยระบบผู้เชี่ยวชาญ จากนั้นเปรียบเทียบกับความถูกต้องของการทำนายโดยโปรแกรมการทำเหมืองข้อมูลแบบจำแนก รวมสามโปรแกรมได้แก่ โปรแกรมที่ใช้หลักการสร้างต้นไม้ตัดสินใจ (ID3) โปรแกรมสร้างกฎการจำแนก (PRISM) และโปรแกรมข่ายงานประสาท (Neural network) ผลการทดสอบยืนยันความถูกต้องในการให้คำแนะนำของระบบผู้เชี่ยวชาญฯ

คำหลัก การทำเหมืองข้อมูล กฎการจำแนก ระบบผู้เชี่ยวชาญ

Abstract

The process of data mining comprises of three major steps: pre-data mining, mining, and post-data mining. This research studies the post-data mining processing. Most data mining systems finish their processing at the knowledge presentation of the mining step. Our work further the post-data mining processing to the step of knowledge deployment. This research illustrates the knowledge deployment step in which its input is the induced knowledge, in the formalism of classification rules. These rules are evaluated and filtered on the basis of coverage measurement. High coverage rules are transformed into decision rules to be used by the inference engine of the expert system. The accuracy of recommendation given by the expert system is evaluated and compared to other three classification systems, i.e. decision-tree induction (ID3), rule induction (PRISM), and neural network. The experimental results confirm the high accuracy of our expert system and the induced knowledge base.

Keywords: data mining, classification rules, expert system

1. บทนำ

การทำเหมืองข้อมูลเป็นเทคโนโลยีใหม่ที่มีจุดมุ่งหมายให้การวิเคราะห์ข้อมูล และการใช้ประโยชน์จากข้อมูลเป็นไปได้อย่างรวดเร็วและสะดวกมากขึ้น แต่อุปสรรคที่สำคัญคือความรู้ที่ค้นพบโดยโปรแกรมทำเหมืองข้อมูล มีจำนวนมาก แต่ความรู้ที่เป็นประโยชน์โดยแท้จริงมีน้อยมาก นอกจากนี้ในบางครั้งความรู้ที่ค้นพบได้มีความซ้ำซ้อนกัน หรือในบางกรณีเป็นความรู้ที่ไม่ใช่การค้นพบใหม่ เช่น "พ่อบ้านในทุกครัวเรือนเป็นเพศชาย" ซึ่งถึงแม้ว่าจะเป็นความรู้ที่ถูกต้องแต่ไม่เกิดประโยชน์ ดังนั้นขั้นตอนสุดท้ายที่สำคัญของการใช้เทคโนโลยีการทำเหมืองข้อมูล คือ การคัดเลือกและกลั่นกรองความรู้ที่ได้ว่า จะนำความรู้ใดรายงานต่อผู้บริหารเพื่อใช้ประโยชน์ในการตัดสินใจ และความรู้ใดควรจะทิ้งไปเพราะซ้ำซ้อนหรือไม่เกิดประโยชน์ ในปัจจุบันกระบวนการพิจารณาและคัดแยกความรู้นี้กระทำโดยผู้เชี่ยวชาญ ที่มีความเข้าใจลักษณะของข้อมูลและมีความสามารถในการตีความผลลัพธ์ที่ได้จากการทำเหมืองข้อมูล ในหน่วยงานหรือองค์กรขนาดใหญ่จึงต้องใช้ทีมบุคลากรที่มีประสบการณ์ด้านการทำเหมืองข้อมูลจำนวนมาก การจะพัฒนาเทคโนโลยีการทำเหมืองข้อมูลให้หน่วยงานขนาดเล็กหรือบุคคลทั่วไปใช้ประโยชน์ได้อย่างทั่วถึง จึงจำเป็นจะต้องมีการพัฒนากระบวนการคัดเลือกความรู้ รวมถึงรวบรวมฐานความรู้ให้มีลักษณะเป็นอัตโนมัติมากขึ้น

การพิจารณาความน่าสนใจ (interestingness) ของความรู้ที่ค้นพบโดยการทำเหมืองข้อมูลเป็นประเด็นปัญหาสำคัญ ที่ได้รับความสนใจจากนักวิจัยจำนวนมากตั้งแต่ยุคเริ่มต้นของงานวิจัยด้านการทำเหมืองข้อมูล เนื่องจากได้เห็นถึงความสำคัญว่าความรู้ที่จะนำไปใช้ประโยชน์ได้ จะต้องเป็นความรู้ที่เป็นการค้นพบใหม่และน่าสนใจ โดย Piatetsky-Shapiro และ Matheus [10] ได้พัฒนาระบบชื่อ KEFIR ขึ้นมาเพื่อใช้กับงานด้านการประกันสุขภาพ ระบบจะพิจารณาข้อมูลที่เบี่ยงเบนไปจากข้อมูลปกติ และตัดสินว่าข้อมูลที่เบี่ยงเบนนี้เป็นข้อมูลที่ควรค่าแก่การให้ความสนใจ การพิจารณาเกณฑ์ความน่าสนใจโดยระบบนี้ จึงเป็นการพิจารณาในขอบเขตที่จำกัดมาก ในช่วงปีต่อมา Silberschatz และ Tuzhilin [12] ได้ร่วมมือกันเสนอวิธีการพิจารณาความน่าสนใจของความรู้ที่ค้นพบ โดยใช้ probabilistic belief เป็นแนวทางในการพิจารณาความน่าจะเป็นว่าความรู้ใดที่ผู้ใช้จะเห็นว่าน่าสนใจ

ปัจจัยที่นักวิจัยใช้พิจารณาประเด็นความน่าสนใจมีได้หลากหลายเช่น จำนวนข้อมูลที่ความรู้นั้นสามารถอธิบายได้ (coverage) ความถูกต้องของความรู้ (confidence) ความไม่แปรผันของความรู้ (strength) ความสำคัญของความรู้ (significance) ความเข้าใจง่ายของความรู้ (simplicity) ความใหม่ของการค้นพบ (unexpectedness) และเป็นความรู้ที่นำไปสู่การปฏิบัติได้ (actionability) ปัจจัยเหล่านี้ถูกหยิบยกขึ้นมาพิจารณาโดยนักวิจัยหลายคณะ [7], [10], [11] ปัจจัยในด้าน coverage, confidence, strength, significance, simplicity เป็นปัจจัยที่สามารถสร้างเทคนิคขึ้นมาจัดการได้ โดยไม่ต้องใช้ข้อมูลหรือคำแนะนำประกอบจากผู้ใช้หรือจากความรู้อื่น แต่ปัจจัยด้าน unexpectedness และ actionability เป็นปัจจัยที่พิจารณาตัดสินได้ยาก

ในส่วนของการจัดการกับปริมาณความรู้ ที่เป็นผลผลิตของกระบวนการทำเหมืองข้อมูล และมักจะได้ความรู้ที่ไม่เป็นประโยชน์ปะปนออกมาเป็นจำนวนมาก เป็นปัญหาที่มีผลกระทบอย่างมากกับงานทำเหมืองข้อมูลโดยเฉพาะงานประเภท association [2], [4], [6], [13] นักวิจัยส่วนใหญ่ [11], [3] จะใช้วิธีจัดการระหว่างกระบวนการทำเหมืองข้อมูล เช่นใช้วิธีตัดทิ้ง (pruning) เส้นทางค้นหาที่คาดว่าจะนำไปสู่ความรู้ที่ไม่เกิดประโยชน์ นักวิจัยในอีกกลุ่ม [5], [8], [9] จะใช้วิธีนำความรู้อื่น (domain knowledge) มาใช้ช่วยในระหว่างการค้นหาความรู้ โดยใช้สมมุติฐานว่าความรู้อื่นๆที่เกี่ยวข้องจะช่วยในการตัดสินใจขจัดทิ้งผลลัพธ์ของการทำเหมืองข้อมูลที่จะไม่ก่อประโยชน์กับงาน

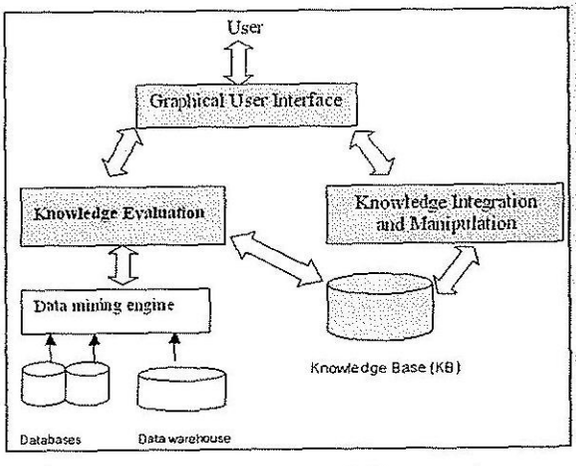
ถึงแม้แนวทางของเทคนิค pruning และเทคนิคการใช้ domain knowledge จะช่วยลดปริมาณความรู้ที่ไม่ก่อประโยชน์ แต่วัตถุประสงค์ของนักวิจัยในทั้งสองกลุ่มนี้มุ่งเน้นไปที่การลดปริมาณความรู้เพื่อเพิ่มประสิทธิภาพของกระบวนการค้นหาความรู้ มากกว่าจะเพื่อเอื้อประโยชน์ให้กับผู้ใช้ ให้สามารถใช้ประโยชน์จากผลลัพธ์ของการทำเหมืองข้อมูลได้อย่างเต็มประสิทธิภาพ และเกิดประโยชน์ในเวลาอันรวดเร็ว ในระยะหลังของงานวิจัยในสาขาการทำเหมืองข้อมูล นักวิจัย [1], [14] เริ่มตระหนักมากขึ้นว่ากระบวนการจัดการกับผลลัพธ์ (หรือ ความรู้) ที่ได้หลังจากการทำเหมืองข้อมูลเป็นสิ่งจำเป็น เพื่อให้การใช้ประโยชน์จากการทำเหมืองข้อมูลเป็นอัตโนมัติและทันต่อความต้องการมากขึ้น

งานวิจัยที่พัฒนาขึ้นนี้ อยู่ในแนวทางของกระบวนการจัดการกับความรู้ ซึ่งจัดเป็น post-processing ของกระบวนการทำเหมืองข้อมูล โดยมีการพัฒนาเทคนิคและกระบวนการในการตรวจสอบความรู้ที่เป็นผลลัพธ์จากการทำเหมืองข้อมูล ให้สามารถแยกแยะความรู้ที่เป็นประโยชน์ออกจากความรู้ที่ใช้ประโยชน์ไม่ได้ จากนั้นเลือกไว้เฉพาะความรู้ที่ผ่านการคัดสรรแล้ว ส่งต่อไปยังฐานความรู้เพื่อใช้ประโยชน์ในงานด้านการสนับสนุนการตัดสินใจ (decision support) การคัดเลือกความรู้ในงานวิจัยนี้ใช้เกณฑ์ coverage เพื่อแปลงเป็นความน่าจะเป็นที่ความรู้นั้นจะสามารถประยุกต์ใช้ได้ โดยประกอบกับค่า threshold ที่ผู้ใช้ระบุเป็นเกณฑ์ขั้นต่ำในการคัดเลือกความรู้

2. กระบวนการประมวลผลหลังการทำเหมืองข้อมูล

2.1 กรอบแนวคิดของงานวิจัย

งานวิจัยนี้มีวัตถุประสงค์หลัก เพื่อการพัฒนาเทคนิคในการจัดการกับความรู้ ที่เป็นผลลัพธ์จากกระบวนการทำเหมืองข้อมูล การจัดการกับความรู้ที่ได้ จึงหมายถึงการประเมินคุณภาพของความรู้เพื่อคัดเลือกเฉพาะความรู้ ที่น่าจะใช้ประโยชน์ได้มากที่สุด โดยระบบต้นแบบที่พัฒนาขึ้น จะประกอบด้วยส่วนประกอบหลักสองส่วนคือ ส่วนวิเคราะห์ความรู้ (knowledge evaluation) ที่ได้จากการทำเหมืองข้อมูล และส่วนรวบรวมและจัดการกับความรู้ (knowledge integration and manipulation) ซึ่งจะรวมถึงการสอบถามความรู้ที่รวบรวมไว้ในฐานความรู้ สถาปัตยกรรมหลักของระบบจัดการความรู้นี้แสดงได้ดังรูปที่ 1



รูปที่ 1 โครงสร้างของระบบจัดการความรู้หลังการทำเหมืองข้อมูล

ส่วนวิเคราะห์ความรู้ ทำหน้าที่รับโมเดลข้อมูลที่เป็นผลลัพธ์จากการทำเหมืองข้อมูลแบบจำแนก โมเดลข้อมูลนี้จะอยู่ในรูปแบบของต้นไม้ตัดสินใจที่มีโครงสร้างหลักคือโหนดต่างๆ (node) ของต้นไม้ และเส้นเชื่อมโหนดทุกเส้น (edge) เมื่อได้รับข้อมูล node และ edge ทั้งหมดจากโปรแกรม data mining engine ส่วนวิเคราะห์ความรู้จะแปลงข้อมูล node และ edge ให้อยู่ในรูปแบบ

ของกฎการตัดสินใจ (decision rule) ในขั้นตอนนี้จะมีการคำนวณค่า coverage ของแต่ละกฎ แล้วแปลงเป็นค่าสัดส่วนของจำนวนข้อมูลที่กฎครอบคลุมเทียบกับจำนวนข้อมูลทั้งหมด ค่าสัดส่วนที่ได้จะมีค่าอยู่ระหว่าง 0.0-1.0 ดังนั้นค่าสัดส่วนนี้สามารถแปลความหมายได้ว่า เป็นความน่าจะเป็นที่กฎจะประยุกต์ใช้ได้กับเหตุการณ์ที่จะเกิดขึ้นในอนาคต ส่วนรวบรวมและจัดการกับความรู้ ทำหน้าที่จัดเก็บความรู้ในรูปแบบของกฎการตัดสินใจ และทำหน้าที่เป็น expert system shell เพื่ออำนวยความสะดวก ให้ผู้ใช้สามารถสอบถามความรู้จากฐานความรู้ได้ ผลลัพธ์ที่แสดงให้ผู้ใช้เห็นจะเป็นข้อแนะนำการตัดสินใจพร้อมกับแสดงค่าความน่าจะเป็น (หรือระดับความเชื่อมั่น) ของข้อแนะนำนั้น

2.2 อัลกอริทึมเพื่อการประเมินและคัดเลือกกฎ

ในการออกแบบอัลกอริทึมที่เป็นโมดูลหลักของโปรแกรม จะประกอบด้วยสองโมดูล คือ Knowledge evaluation และ Knowledge integration and manipulation รายละเอียดของแต่ละอัลกอริทึมแสดงได้ดังต่อไปนี้

Algorithm 1 Knowledge evaluation
Input: a data model as decision tree with node and edge structures
Output: a set of probabilistic decision rules ranking in descending order
(1) Display GUI to get a dataset name and a minimum probability value
(2) Traverse tree from a root node to each leaf node
  (2.1) Collect edge information and count number of data instances
  (2.2) Compute probability as a proportion (number of instances at leaf node) / (total data instances in a data set)
  (2.3) Assert a rule containing a triplet <attribute-value pair, class, probability value> into temporary KB
(3) Sort rules in the KB in descending order according to the rules' probability
(4) Remove rules that have probability less than the specified threshold
(5) Assert selected rules into the KB and return KB as an output

Algorithm 2 Knowledge integration and manipulation
Input: a set of probabilistic decision rules stored in KB
Output: a set of rules to be used by an expert system shell
(1) For each probabilistic decision rule
  (1.1) Scan information in the If-part and the Then-part
  (1.2) Generate head of expert rule from the Then-part type (Then-part, probability value) :-
  (1.3) Generate body of expert rule from the If-part :- atrribute_name1(value), …, attribute_nameN(value).
  (1.4) Write expert rule in a file, knb-file
(2) For each data attribute
  (2.1) Generate head of consulting rule attribute_name(X) :-
  (2.2) Generate body of consulting rule :- menuask(attribute_name, X, [list of attribute values]).
(3) Assert consulting rules into the knb-file and return KB as an output

3. การพัฒนาโปรแกรมและการประมวลผล

โปรแกรมนี้พัฒนาด้วยภาษาโปรล็อก การอธิบายขั้นตอนพัฒนาโปรแกรมจะใช้ไวยากรณ์ของภาษาโปรล็อก ตามมาตรฐานของ SWI Prolog เวอร์ชัน 5.6.55 (ดาวน์โหลดได้จากเว็บไซต์ www.swi-prolog.org) ลักษณะเด่นประการหนึ่งของภาษาโปรล็อกคือการใช้รูปแบบเดียวกันของทั้งข้อมูลและคำสั่งที่ทำงานกับข้อมูล

รูปแบบของข้อมูล

ข้อมูลตัวอย่างที่จะเป็นอินพุทของโปรแกรมทำเหมืองข้อมูลจะมีลักษณะเป็นข้อความที่อยู่ในรูปแบบของข้อความที่เป็นจริงหรือ fact ตัวอย่างของข้อมูลแสดงได้ดังรูปที่ 2 ข้อมูลดังรูปเป็นข้อมูลการวินิจฉัยของจักษุแพทย์ว่าคนไข้จำนวน 24 รายที่มารับการวินิจฉัยแต่ละรายสามารถใส่คอนแท็กเลนส์ได้หรือไม่ ถ้าคนไข้สามารถใส่คอนแท็กเลนส์ได้จะมีค่า class=yes ส่วนรายที่ไม่สามารถใส่คอนแท็กเลนส์ (เนื่องจากพยาธิสภาพไม่เหมาะสม เช่น มีนัยน์ตาแห้งเกินไป) จะมีค่า class=no

```
%% Data lens
%   attributes: names and their possible values
attribute(age, [young, pre_presbyopic, presbyopic]).
attribute(spectacle, [myope, hypermetrope]).
attribute(astigmatism, [no, yes]).
attribute(tear, [reduced, normal]).
attribute(class, [ yes, no]).
%   data
instance(1, class=no, [age=young, spectacle=myope,
                       astigmatism=no, tear=reduced]).
instance(2, class=yes, [age=young, spectacle=myope,
                       astigmatism=no, tear=normal]).
instance(3, class=no, [age=young, spectacle=myope,
                       astigmatism=yes, tear=reduced]).
instance(4, class=yes, [age=young, spectacle=myope,
                       astigmatism=no, tear=normal]).
...
```

รูปที่ 2 ข้อมูลตัวอย่างของผู้ที่ได้รับการตรวจก่อนใส่คอนแท็กเลนส์

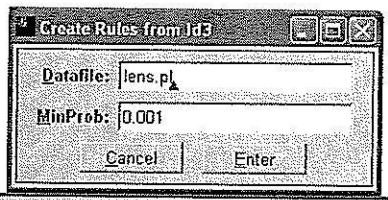
บรรทัดแรกของข้อมูลเริ่มต้นด้วยเครื่องหมาย % หมายถึง comment โครงสร้างของไฟล์ข้อมูลจะแยกส่วนประกอบออกเป็นสองส่วนคือ ส่วนคำอธิบายแอททริบิวต์ (ได้แก่ส่วนข้อความ attribute ...) และส่วนแสดงรายละเอียดของข้อมูลแต่ละเรคคอร์ด (ได้แก่ส่วนข้อความ instance ...) ในส่วนที่อธิบายแอททริบิวต์ภายในจะประกอบด้วยสองอาร์กิวเม้นต์ อาร์กิวเม้นต์แรกจะบอกชื่อแอททริบิวต์ อาร์กิวเม้นต์ที่สองเป็นลิสต์ของค่าที่เป็นไปได้ทั้งหมดของแอททริบิวต์นั้น ในส่วนของข้อมูลหรือ instance จะประกอบด้วยสามอาร์กิวเม้นต์ คือ หมายเลขของข้อมูล, ค่าคลาสของข้อมูล, ลิสต์ที่ระบุค่าของแต่ละแอททริบิวต์ โดยวิธีการระบุค่าจะใช้รูปแบบ attribute-value pair หรือ ชื่อแอททริบิวต์=ค่าของแอททริบิวต์ เมื่อสร้างข้อมูลในรูปแบบที่กำหนดเสร็จ จะต้องบันทึกไฟล์ให้อยู่ในรูปแบบของโปรแกรม Prolog ที่มีส่วนขยาย (file extension) เป็น .pl เช่นตัวอย่างไฟล์ข้อมูลในรูปที่ 2 บันทึกอยู่ในชื่อ lens.pl

โปรแกรมวิเคราะห์ความรู้

การทำงานของโปรแกรมในส่วนนี้ จะเริ่มหลังจากมีการสร้างโมเดลข้อมูลด้วยหลักการของการสร้างต้นไม้ตัดสินใจ (อัลกอริทึม ID3 [11]) และบันทึกโมเดลไว้ในเพรดิเคตชื่อ node และ edge คำสั่งต่างๆ ของโปรแกรมหลักแสดงได้ดังนี้

```
mainId3(Min) :-
    init(AllAttr, EdgeList),   % initialize node and edge info.
    getnode(N),                % get node ID
    create_edge(N, AllAttr, EdgeList),  % create tree
    addAllKnowledge,           % generate decision rules
    selectRule(Min, Res),      % select top rules
    tell('1.knb'),             % write selected decision rules to file
    writeHeadF,                % transform to expert rules (head)
    maplist(createRule1, Res),
    nl, writeTailF,            % generate body of expert rules
    told,                      % write expert rules to file and close it
    writeln(endProcess).
```

โปรแกรมหลักชื่อ mainId3 จะรับอาร์กิวเม้นต์ Min ที่ระบุค่าความน่าจะเป็นขั้นต่ำที่ผู้ใช้ต้องการ จากนั้นเริ่มสร้างต้นไม้ตัดสินใจด้วยการเรียกใช้คำสั่งย่อยสามคำสั่งคือ init, getnode และ create_edge คำสั่ง create_edge จะมีการวนทำงานซ้ำเพื่อสร้างต้นไม้ตัดสินใจ การสร้างต้นไม้จะยุติเมื่อสามารถแยกข้อมูลที่มีสองคลาสปนกัน ให้เป็นข้อมูลคลาสเดียวกันได้ทั้งหมด หรือเมื่อไม่มีแอททริบิวต์ให้ใช้แยกข้อมูลได้อีกต่อไป จากข้อมูล lens.pl ในรูปที่ 2 เมื่อสร้างต้นไม้ตัดสินใจเสร็จ จะได้โครงสร้าง node และ edge ดังรูปที่ 3




```
File  Edit  Settings  Run  Debug  Help
1 ?- listing(node).
:- dynamic node/2.

node(1, [2, 4, 6, 8, 10, 12, 14, 20, 22]-[1, 3, 5, 7
, 9, 11, 13, 15, 16, 17, 18, 19, 21, 23, 24]).
node(2, []-[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 2
3]).
node(3, [2, 4, 6, 8, 10, 12, 14, 20, 22]-[16, 18, 24
]).
node(4, [2, 4, 6, 8]-[]).
node(5, [10, 12, 14]-[16]).
node(6, [10, 12]-[]).
node(7, [14]-[16]).
node(8, [20, 22]-[18, 24]).
node(9, [20]-[18]).
node(10, [22]-[24]).

true.

2 ?- listing(edge).
:- dynamic edge/3.

edge(0, root=nil, 1).
edge(1, tear=reduced, 2).
edge(1, tear=normal, 3).
edge(3, age=young, 4).
edge(3, age=pre_presbyopic, 5).
edge(5, spectacle=myope, 6).
edge(5, spectacle=hypermetrope, 7).
edge(3, age=presbyopic, 8).
edge(8, spectacle=myope, 9).
edge(8, spectacle=hypermetrope, 10).

true.
```

รูปที่ 3 จอภาพแสดงการระบุชื่อข้อมูลและผลลัพธ์เป็นโมเดลข้อมูลในลักษณะโครงสร้าง node และ edge ของต้นไม้ตัดสินใจ

ตัวอย่างในรูปที่ 3 ระบุค่าความน่าจะเป็นขั้นต่ำ 0.001 จะทำให้ได้กฎการตัดสินใจจำนวนสามกฎ ดังต่อไปนี้

```
0.5 >> [tear=reduced] >> no,
0.166667 >> [tear=normal, age=young] >> yes,
0.0833333 >> [tear=normal, age=pre_presbyopic,
                spectacle=myope] >> yes
```

โครงสร้างของกฎการตัดสินใจจะประกอบด้วยสามส่วน แต่ละส่วนแยกจากกันด้วยสัญลักษณ์ >> ส่วนแรกระบุความน่าจะเป็นที่กฎนี้สามารถครอบคลุมข้อมูล ส่วนที่สองระบุลักษณะหรือแอททริบิวต์ที่ใช้ประกอบการตัดสินใจ และส่วนสุดท้ายเป็นผลการวินิจฉัยของแพทย์ ค่า yes หมายถึงแนะนำให้คนไข้ที่มีปัญหาด้านสายตาใส่คอนแท็กเลนส์ได้ ค่า no หมายถึงไม่แนะนำให้ใส่คอนแท็กเลนส์ ตัวอย่างเช่นกฎข้อแรกระบุว่า ถ้าคนไข้มีอัตราการสร้างน้ำตาต่ำ (tear=reduced) แพทย์จะไม่แนะนำให้ใส่คอนแท็กเลนส์ ค่าความน่าจะเป็น 0.5 หมายถึงกฎนี้ครอบคลุมข้อมูล 0.5 หรือ 50% (นั่นคือกฎนี้สังเคราะห์ได้จากข้อมูลคนไข้ 12 รายจากข้อมูลคนไข้ทั้งหมดจำนวน 24 ราย)

โปรแกรมรวบรวมและจัดการกับความรู้

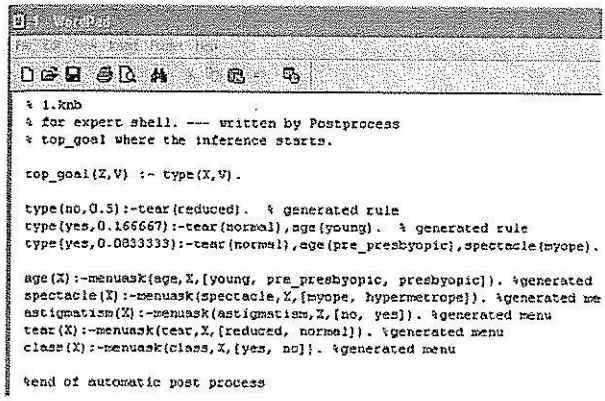
กฎการตัดสินใจที่ได้จากโปรแกรมวิเคราะห์ความรู้ จะถูกนำมาแปลงให้เป็นกฎเพื่อใช้ในระบบผู้เชี่ยวชาญ โปรแกรมแปลงกฎการตัดสินใจให้เป็นกฎประเภท expert rules แสดงได้ดังนี้

```
writeHeadF :-
        format('% 1.knb ~n% for expert shell. --- written
                by Postprocess'),
        format('~n% top_goal where the inference
                starts.~n'),
        format('~ntop_goal(X,V) :- type(X,V).~n').

writeTailF:-
        findall(_,(attribute(S,L),
        format('~n~w(X):-menuask(~w,X,~w).
                %generated menu',[S,S,L])) ,_),
        format('~n~n%end of automatic post process').
```

โปรแกรมแปลงกฎการตัดสินใจให้เป็นกฎประเภท consult rules แสดงได้ดังนี้

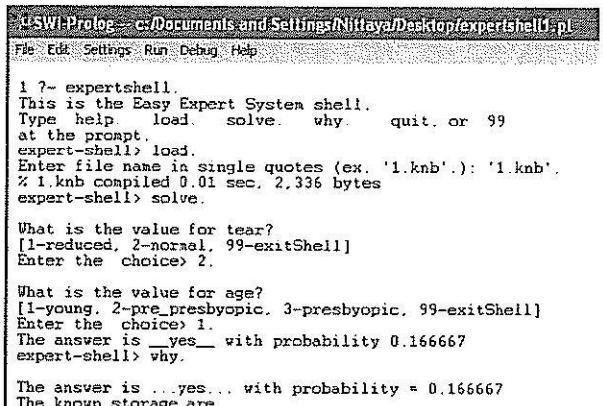
```
transform1([X=V], [Res]) :-
        atomic_list_concat([X,'(',V,')'], Res1),
        term_to_atom(Res, Res1),!.

transform1([X=V|T], [Res|T1]) :-
        atomic_list_concat([X,'(',V,')'], Res1),
        term_to_atom(Res, Res1),
        transform1(T, T1).

createRule1(I) :- I=Z>>X>>Y,
        transform1(X,BodyL),
        format('~ntype(~w,~w):-',[Y,Z]),
                myformat(BodyL) ,
        write(' % generated rule'),!.

myformat([X]) :- write(X), write('.'),!.
myformat([H|T]) :- write(H), write(','), myformat(T).
```

เมื่อโปรแกรมที่ทำหน้าที่แปลงจากกฎการตัดสินใจ ให้เป็นกฎที่สามารถใช้งานได้กับระบบผู้เชี่ยวชาญ ทำการแปลงกฎทั้งหมดเสร็จสิ้น จะบันทึกผลลัพธ์ที่ได้ลงในไฟล์ชื่อ '1.knb' ซึ่งจากตัวอย่างข้อมูล lens.pl เมื่อแปลงกฎเสร็จจะได้ข้อมูลดังรูปที่ 4 ซึ่งจะทำหน้าที่เป็นฐานความรู้ให้กับระบบผู้เชี่ยวชาญ ตัวอย่างของการใช้งานระบบผู้เชี่ยวชาญแสดงได้ดังรูปที่ 5



```
% 1.knb
% for expert shell. --- written by Postprocess
% top_goal where the inference starts.

top_goal(X,V) :- type(X,V).

type(no,0.5):-tear(reduced).  % generated rule
type(yes,0.166667):-tear(normal),age(young).  % generated rule
type(yes,0.0833333):-tear(normal),age(pre_presbyopic),spectacle(myope).

age(X):-menuask(age,X,[young, pre_presbyopic, presbyopic]). %generated
spectacle(X):-menuask(spectacle,X,[myope, hypermetrope]). %generated me
astigmatism(X):-menuask(astigmatism,X,[no, yes]). %generated menu
tear(X):-menuask(tear,X,[reduced, normal]). %generated menu
class(X):-menuask(class,X,[yes, no]). %generated menu

%end of automatic post process
```

รูปที่ 4 ข้อมูลในไฟล์ที่ทำหน้าที่เป็นฐานความรู้ของระบบผู้เชี่ยวชาญ



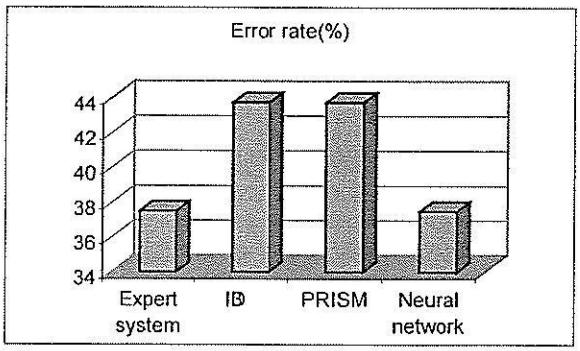
```
SWI-Prolog -- c:/Documents and Settings/Nittaya/Desktop/expertshell1.pl
File  Edit  Settings  Run  Debug  Help

1 ?- expertshell.
This is the Easy Expert System shell.
Type  help.   load.   solve.   why.    quit. or  99
at the prompt.
expert-shell> load.
Enter file name in single quotes (ex. '1.knb'.): '1.knb'.
% 1.knb compiled 0.01 sec, 2,336 bytes
expert-shell> solve.

What is the value for tear?
[1-reduced, 2-normal, 99-exitShell]
Enter the  choice> 2.

What is the value for age?
[1-young, 2-pre_presbyopic, 3-presbyopic, 99-exitShell]
Enter the  choice> 1.
The answer is __yes__ with probability 0.166667
expert-shell> why.

The answer is ...yes... with probability = 0.166667
The known storage are
[age(young), tear(normal)]
expert-shell>
```

รูปที่ 5 ตัวอย่างการใช้งานระบบผู้เชี่ยวชาญเพื่อขอคำแนะนำเกี่ยวกับการใช้คอนแท็กเลนส์

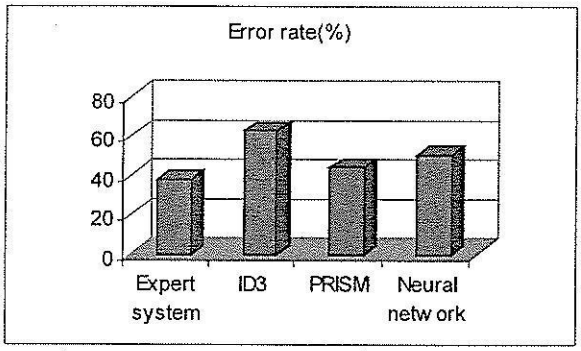
4. การทดสอบฐานความรู้

วิธีการทดสอบการสร้างโมเดลของโปรแกรมเพื่อการประมวลผลหลังการทำเหมืองข้อมูล จะใช้วิธีแยกข้อมูลส่วนใหญ่เป็นข้อมูลฝึก เพื่อสร้างโมเดลข้อมูลและสร้างกฎการตัดสินใจที่จะถูกแปลงเป็นกฎในระบบผู้เชี่ยวชาญ จากนั้นจะทดสอบความถูกต้องในการให้คำแนะนำของระบบผู้เชี่ยวชาญ โดยข้อมูลที่ใช้ทดสอบเป็นข้อมูลขนาดเล็กมีจำนวน 16 รายการ ชุดข้อมูลที่ใช้เป็นข้อมูลมาตรฐานของ UCI (http://archive.ics.uci.edu/ml/datasets.html) โดยเลือกใช้ข้อมูล post-operative patients (จำนวนข้อมูลฝึกมี 70 เรคคอร์ด แต่ละเรคคอร์ดประกอบด้วย 8 แอททริบิวต์) และข้อมูล breast-cancer recurrences (จำนวนข้อมูลฝึกมี 175 เรคคอร์ด แต่ละเรคคอร์ดประกอบด้วย 9 แอททริบิวต์)

การทดสอบในประเด็นความถูกต้องในการให้คำแนะนำของระบบผู้เชี่ยวชาญที่พัฒนาขึ้น จะเปรียบเทียบกับผลการทดสอบในการจำแนกข้อมูลด้วยอัลกอริทึม ID3 (decision-tree induction algorithm), PRISM (rule induction algorithm) และ neural network (multi-layer perceptron algorithm) ผลการเปรียบเทียบความผิดพลาดในการให้คำแนะนำ หรือการทำนายคลาสของข้อมูลทดสอบของโปรแกรม expert system, ID3, PRISM, Neural network กับข้อมูล post-operative patients และ breast-cancer recurrences แสดงดังกราฟในรูปที่ 6 และ 7 ตามลำดับ



รูปที่ 6 กราฟเปรียบเทียบ error rate เมื่อทดสอบกับข้อมูล post-operative patients



รูปที่ 7 กราฟเปรียบเทียบ error rate เมื่อทดสอบกับข้อมูล breast-cancer recurrences

จากผลการทดสอบความถูกต้องของโปรแกรมประมวลผลหลังการทำเหมืองข้อมูล หรือโปรแกรม automatic expert system ที่สร้างระบบผู้เชี่ยวชาญโดยอัตโนมัติ ในทั้งสองชุดข้อมูลเมื่อเทียบกับโปรแกรม ID3 พบว่าโปรแกรม automatic expert system ให้ผลลัพธ์ที่ถูกต้องมากกว่าทั้งๆที่โปรแกรม automatic expert system ใช้โมเดลข้อมูลเริ่มต้นเป็นต้นไม้ตัดสินใจเหมือนกับโปรแกรม ID3 แต่ข้อแตกต่างอยู่ที่ โปรแกรม automatic expert system มีการจัดลำดับกฎการตัดสินใจตามค่าความน่าจะเป็นและคัดเลือกใช้เฉพาะกฎที่มีค่าความน่าจะเป็นสูง

เมื่อเปรียบเทียบความผิดพลาดในการให้คำแนะนำ หรือความผิดพลาดในการทำนาย (error rate) ของโปรแกรมทั้งสี่พบว่าโปรแกรม automatic expert system ให้ค่าความผิดพลาดที่ต่ำที่สุด (คือ 37.50% ในข้อมูลทั้งสองชุด) และเมื่อวิเคราะห์ความผิด

พลาดในลักษณะของ false negative พบว่าโปรแกรม automatic expert system ให้ความผิดพลาดชนิดนี้ต่ำที่สุด ซึ่งในการวินิจฉัยทางด้านการแพทย์ความผิดพลาดประเภท false negative (เช่นคนไข้เป็นมะเร็ง แต่ได้รับการวินิจฉัยว่าสุขภาพแข็งแรงเป็นปกติ) ถือว่ามีความร้ายแรงมากกว่าความผิดพลาดประเภท false positive (เช่นคนไข้ที่ไม่เป็นมะเร็ง แต่ได้รับการวินิจฉัยว่าเป็นมะเร็งและจะต้องถูกส่งไปตรวจทางคลินิกเพิ่มเติม)

โปรแกรมทำเหมืองข้อมูลที่ใช้โมเดลข้อมูลในลักษณะของต้นไม้ตัดสินใจ (ID3) หรือในลักษณะของกฎ (automatic expert system และ PRISM) จะมีข้อบกพร่องที่เห็นได้ชัดเจนจากผลการทดลองคือ โมเดลที่ได้ไม่สมบูรณ์ เนื่องจากจะมีบางกรณีที่ไม่สามารถให้คำแนะนำ หรือไม่สามารถทำนายคลาสของข้อมูลได้ ซึ่งจะต่างจากโปรแกรม Neural network ที่มีพื้นฐานการสร้างโมเดลจากการใช้ฟังก์ชันทางคณิตศาสตร์ (ในการทดลองนี้ใช้ sigmoid function) ทำให้สามารถทำนายคลาสของข้อมูลได้ครอบคลุมหมดทุกกรณี

5. สรุป

โปรแกรมเพื่อการทำเหมืองข้อมูลในปัจจุบัน มักจะได้รับการพัฒนาโปรแกรมจนถึงขั้นให้สามารถแสดงผลลัพธ์เป็นโมเดลข้อมูลหรือแพทเทิร์นของกลุ่มข้อมูล ขั้นตอนต่อจากนั้นที่เป็นการนำโมเดลหรือแพทเทิร์นไปใช้ประโยชน์ มักจะให้อยู่ในดุลพินิจของนักวิเคราะห์ข้อมูล ซึ่งนักวิเคราะห์ข้อมูลที่ไม่คุ้นเคยกับกระบวนการทำเหมืองข้อมูล มักจะประสบกับอุปสรรคสำคัญของการคัดเลือกความรู้ในโมเดลไปใช้งานต่อไป เนื่องจากโปรแกรมทำเหมืองข้อมูลมักจะแสดงผลลัพธ์เป็นโมเดลที่มีความซับซ้อนและมีขนาดใหญ่ทำให้แปลผลได้ยาก งานวิจัยเรื่องการประมวลผลหลังกระบวนการทำเหมืองข้อมูลนี้ จึงมีวัตถุประสงค์หลักเพื่อจะพัฒนาโปรแกรมเพื่อรับอินพุทเป็นโมเดลข้อมูล จากนั้นแปลงโมเดลข้อมูลให้อยู่ในรูปแบบที่นักวิเคราะห์ข้อมูลโดยทั่วไปเข้าใจได้

นอกจากแปลงรูปแบบโมเดลข้อมูลให้เข้าใจได้ง่ายแล้ว งานวิจัยนี้ยังได้เพิ่มเทคนิคของการกลั่นกรองโมเดล ให้มีขนาดเล็กลง ทั้งนี้เพื่อความสะดวกในการแปลผลและนำไปใช้งาน โปรแกรมที่พัฒนาขึ้นมีฟังก์ชันด้านการใช้งานโมเดลที่ได้รับการกลั่นกรองแล้วด้วยการจัดสร้างฐานความรู้โดยอัตโนมัติจากโมเดลข้อมูล และสร้าง expert system shell ที่ผู้ใช้สามารถสอบถามโมเดลข้อมูลที่อยู่ในฐานความรู้ได้อย่างสะดวก

กิตติกรรมประกาศ

งานวิจัยนี้ได้รับการสนับสนุนจากมหาวิทยาลัยเทคโนโลยีสุรนารี ผ่านการดำเนินงานของหน่วยวิจัยวิศวกรรมข้อมูลและการค้นหาความรู้ (DEKD Research Unit) และสำนักงานคณะกรรมการวิจัยแห่งชาติ (วช) ผู้วิจัยรายชื่อที่สองได้รับการสนับสนุนทุนวิจัยจากสำนักงานกองทุนสนับสนุนการวิจัย (สกว. สัญญาเลขที่ RMU 5080026)

เอกสารอ้างอิง

[1] Adomavicius, G. and Tuzhilin, A. 2001. Expert-driven validation of rule-based user models in personalization applications. *Journal of Data Mining and Knowledge Discovery*, 5(1/2): 33-58.

[2] Bayardo, R.J., Agrawal, R. and Gunopulos, D. 1999. Constraint-based rule mining in large, dense databases. *Proc. of the 15th Int. Conf. on Data Engineering*, March.

[3] Breiman, L., Friedman, J., Olshen, R. and Stone, C. 1984. *Classification and regression tree*. Belmont: Wadsworth.

[4] Brin, S., Motwani, R., Ullman, J.D. and Tsur, S. 1997. Dynamic itemset counting and implication rules for market basket data. *Proc. of the ACM SIGMOD Conference*.

[5] Clark, P. and Matwin, S. 1993. Using qualitative models to guide induction learning. *Proc. of Int. Conf. on Machine Learning*.

[6] Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H. and Verkamo, A.I. 1994. Finding interesting rules from large sets of discovered association rules. *Proc. of the Third Int. Conf. on Information and Knowledge Management*.

[7] Major, J. and Mangano, J. 1993. Selecting among rules induced from a hurricane database. *Proc. of the AAAI-93 Workshop on KDD*.

[8] Ortega, J. and Fisher, D. 1995. Flexibly exploiting prior knowledge in empirical learning. *IJCAI*.

[9] Pazzani, M. and Kibler, D. 1992. The utility of knowledge in inductive learning. *Machine Learning*, 9.

[10] Piatetsky-Shapiro, G. and Matheus, C.J. 1994. The interestingness of deviations. *Proc. of the AAAI-94 Workshop on Knowledge Discovery in Databases*.

[11] Quinlan, J.R. 1992. *C4.5: Program for Machine Learning*. Morgan Kaufmann.

[12] Silberschatz, A. and Tuzhilin, A. 1996. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(6), December.

[13] Suzuki, E. 1997. Autonomous discovery of reliable exception rules. *Proc. of the Third Int. Conf. on Knowledge Discovery and Data Mining*.

[14] Wang, K., Tay, S.H.W and Liu, B. 1998. Interestingness-based interval merger for numeric association rules. *Proc. of the Fourth Int. Conf. on Knowledge Discovery and Data Mining*.

Probabilistic Knowledge Discovery from Medical Databases

Nittaya Kerdprasop and Kittisak Kerdprasop
Data Engineering and Knowledge Discovery (DEKD) Research Unit,
School of Computer Engineering, Suranaree University of Technology,
111 University Avenue, Nakhon Ratchasima 30000 Thailand
E-mail: nittaya@sut.ac.th; **Fax**: 044-224602; **Tel**. 044-224432

ABSTRACT

Medical knowledge discovery is an emerging area within the data-mining field that attracts many new researchers from diverse disciplines. During the past decade numerous learning techniques had been employed to discover useful knowledge from health examination and clinical data. Unlike past efforts that simply concentrated on the deployment of well-known learning techniques on medical data sets, our new approach expands the learning algorithm to deal with uncertain knowledge. We devise an algorithm to generate probabilistic knowledge from the induced decision tree. The implementation of the proposed algorithm is demonstrated via second-order predicates and a meta-programming approach. Experimental results on several medical domains emphasize the simple form of knowledge representation and the potential of incorporating learning results as background knowledge in the knowledge-base system.

Keywords: Medical knowledge mining, Probabilistic knowledge induction.

1. INTRODUCTION

The automated learning of models from patient data and biomedical records has become more and more essential since the extensive computerization in healthcare industry and the significant advancement in genomic and proteomic technologies during the last decade. Medical and clinical databases have been created and constantly growing at an exponential rate. The development of an automatic and intelligent data analysis tool is an obvious solution to the data-flooding problem in medical domains [4], [12], [13].

Knowledge extraction from huge amount of health databases is expected to ease the medical decision-making process. The ultimate goal of knowledge extraction is to generate the most accurate and useful knowledge and represent it in an understandable format. Such goal is, however, difficult to accomplish due to the learning complexity of knowledge induction methods and the nonconformity of the database contents. Most of the time knowledge discovery from medical databases results in reporting large number of irrelevant knowledge [6], [9]. We thus focus our study on this issue and devise a technique to extract a limited number of knowledge that is most likely relevant to the specific domain.

In medical knowledge mining, interpretability of results is an important feature of the data analysis tool. Medical practitioners need a system that can produce accurate results in an understandable form. Therefore, knowledge represented as rules has been widely used for knowledge discovery in medical applications. Classification rule induction is an approach commonly used for building diagnosis models [2], [5], [7], [14]. Association rule mining is an induction method applied for exploring patterns that are frequently occur in medical data [3], [10]. Classification and association mining methods are two major techniques for rule generation that work successfully in many domains. Nevertheless, in medical applications these learning techniques tend to generate a lot of rules. Too many rules, some are redundant and uninteresting, cause problems to the medical practitioners because a truly relevant one can be easily overlooked.

We thus propose a rule induction method based on the decision-tree structure that adopts the probability concept to select the most probable applicable rules. The outline of this paper is as follows. After the introductory section, we present in Section 2 the general framework and the algorithms of our proposed method. The implementation and experimental results are illustrated in Sections 3. Section 4 concludes this paper with the discussion on further research direction.

2. FRAMEWORK AND METHOD FOR PROBABILISTIC KNOWLEDGE INDUCTION

Our knowledge induction system (Figure 1) is based on the decision-tree induction method [11]. Decision tree induction is a popular method for mining knowledge from data and representing the result as a classifier tree. Popularity is due to the fact that mining result in a form of decision tree is interpretability, which is more concern among casual users than a sophisticated method but lack of understandability. A decision tree is a hierarchical structure with each node contains decision attribute and node branches corresponding to different attribute values of the decision node. The goal of building decision tree is to partition data with mixing classes down the tree until each leaf node contains data with pure class.

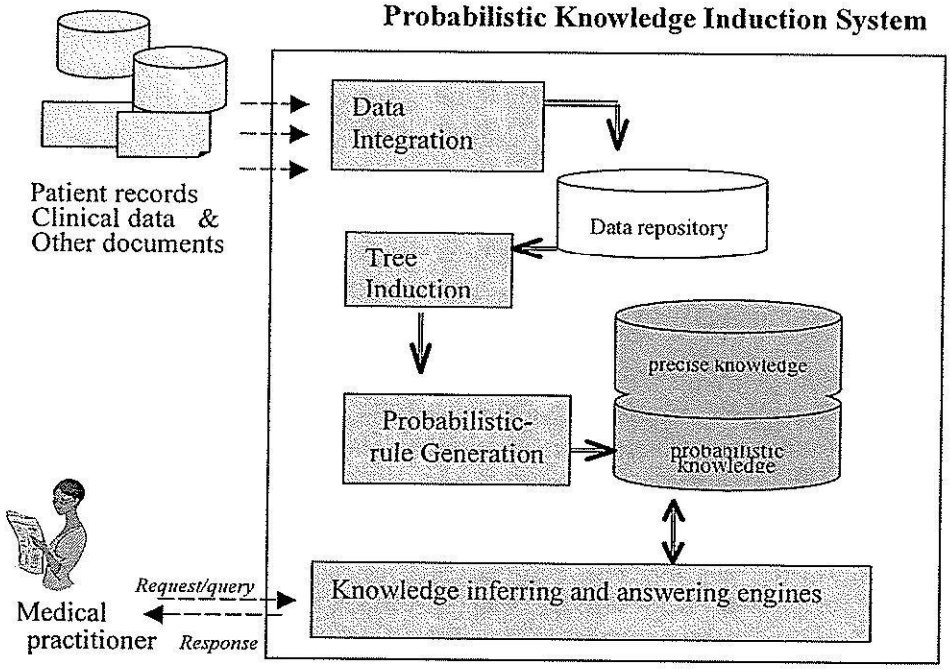


Figure 1. A general framework for the tree-based probabilistic knowledge induction.

In our system framework, we increase interpretability of the knowledge mining results by transforming the decision tree structure into a small set of decision rules. After a complete decision tree has been created, we calculate the probability of case occurrence augmented with each leaf node. In the phase of decision rule generation, these probability values will be sorted. Rules within the top ranking part will be displayed to assist medical practitioner for making decision. In the designed framework, probabilistic knowledge induction system is composed of four main components: data integration, tree induction, probabilistic-rule generation, and the knowledge inferring and answering engines. Data integration component is responsible for collecting data from different sources, cleaning and format transforming. These data are to be used by the tree induction component.

Given the induced tree, the probabilistic-rule generation component traverses each tree branch to calculate the likelihood of path occurrence. This likelihood is interpreted as the probability of event and associated to the rule generated from the path traversal. The generated probabilistic rules are then sorted. Rules at the top ranking (specified by the given minimum probability) are stored in the knowledge base as the probabilistic knowledge and could be used for recommendation or answering query to the medical practitioner. Algorithms for knowledge induction based on tree structure (Algorithm 1), probabilistic-rule generation from decision tree (Algorithm 2), and probabilistic knowledge inferring to answer the most probable class decision on new case (Algorithm 3) are given in the following.

Algorithm 1 Knowledge induction
Input: a data set formatted as Prolog clauses
Output: a decision tree with node and edge structures
(1) Initialization
  (1.1) Clear temporary knowledge base (KB) by removing all information regarding the predicates node, edge and current_node
  (1.2) Set node counter = 0
  (1.3) Scan data set to get information about data attributes, positive instances, negative instances, total data instances
(2) Building tree
  (2.1) Increment node counter
  (2.2) Repeat steps 2.2.1-2.2.4 until there is no more attributes left for creating decision attributes
    (2.2.1) Compute Info value of each candidate attribute
    (2.2.2) Choose the attribute that yields minimum Info to be decision node
    (2.2.3) Assert edge and node information into KB
    (2.2.4) Split data instances along node branches
  (2.3) Repeat steps 2.1 and 2.2 until the lists of positive and negative instances are empty
  (2.4) Output tree structure containing node and edge predicates

Algorithm 2 Probabilistic knowledge generation
Input: a decision tree with node and edge structures, and a probability threshold
Output: a set of probabilistic rules ranking from the highest probability

(1) Traverse tree from a root node to each leaf node
  (1.1) Collect edge information and count number of data instances
  (1.2) Compute probability as a proportion
    (number of instances at leaf node) / (total data instances in a data set)
  (1.3) Assert a rule containing a triplet (attribute-value pair, class, probability value) into KB
(2) Sort rules in the KB in descending order according to the rules' probability
(3) Remove rules that have probability less than the specified threshold
(4) Assert selected rules into the KB and return KB as an output

Algorithm 3 Probabilistic knowledge inferring
Input: a KB containing probabilistic knowledge, and a new case with unknown class value
Output: a decision on most likely class of the new case
(1) Read all attribute-value pairs appeared in the given case
(2) Compare the pairs with each relevant rule in the KB to get the decision class value
(3) Compute the decision confidence as
  (number of matched attribute-value pair) × (probability of the decision rule)
(4) Output a final decision based on the voting scheme

3. IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this section, we present the implementation technique of our proposed tree-based probabilistic-knowledge induction framework using logic programming paradigm. Prolog code is based on the syntax of SWI Prolog (www.swi-prolog.org).

Data format. In logic programming, program and data take the same format, i.e. all are in Prolog clausal form. For the purpose of demonstration, we use the health examination data of 86 patients after the operation. The general conditions such as blood pressure and temperature are observed to determine whether the patient is in good condition and should be sent home shortly (class=home), or the condition is quite moderate and should stay at the hospital ward for further follow up (class=ward). Even though binary classification is a typical task in medical domains, the code presented in this section can be easily modified to classify data with more than two classes. The post-operative data set (downloadable from the UCI repository [1]) in Prolog clausal form is shown some part as the following.

```
attribute( internalTemp,      [mid, high, low]).
attribute( surfaceTemp,       [mid, high, low]).
attribute( oxygenSaturation,  [excellent, good, fair, poor]).
attribute( bloodPressure,     [high, mid, low]).
attribute( tempStability,     [stable, mod-stable, unstable]).
attribute( coreTempStability, [stable, mod-stable, unstable]).
attribute( bpStability,       [stable, mod-stable, unstable]).
attribute( comfort,           [5,7,10,15]).
attribute( class,             [home, ward]).
instance(1,class=ward,[internalTemp=mid, surfaceTemp=low, oxygenSaturation=excellent,
                       bloodPressure=mid, tempStability=stable, coreTempStability=stable,
                       bpStability=stable, comfort=15]).
```

Main module. The three algorithms, explained in the previous section, are called by the main module, which is the top-level of our program implementation. The Prolog coding of main module is as follows:

```
main:-init(AllAttr,EdgeList), getnode(N),
      create_edge_onelevel(N,AllAttr,EdgeList),
      addKnowledge,
      write(chooseMinProb),read(Min),
      selectRule(Min,Res), maplist(writeln,Res).
```

The built-in predicate `maplist` is a second-order predicate [8] provided in the library of SWI Prolog. Its implementation is declared recursively as the following. The predicates `init` and `getnode` in main module invoke the following initialization process. The predicates `assert` and `retractall` are also second-order predicates responsible for asserting and removing, respectively, information in the knowledge base. Another second-order predicate `apply` repeatedly assert clauses into the knowledge base. The built-in second-order predicate `findall` searches for all solutions that satisfy the constrained predicates.

Probabilistic-rule generation. In main module, the predicates `addKnowledge` and `selectRule(Min,Res)` are invoked to compute probability along each tree branch to generate probabilistic rules and then select only rules that could occur at the probability level higher than the specified threshold. Prolog coding of this module is as follows:

```
addKnowledge:- findall([A],pathFromRootToLeaf(A,_),Res),
               retractall(_>>_>>_), maplist(apply(assert),Res).
```

```
selectRule(V,Res):- findall(N>>X>>Class,(X>>Class>>N,N>=V),Res1),
                    sort(Res1,Res2), reverse(Res2,Res).
pathFromRootToLeaf(V>>Class>>Num,C):-   path(1,V,C),
                node(C,Value1-Value2),(Value1=[]; Value2=[]),
                (Value1=[]->length(Value2,Numb); length(Value1,Numb)),
                total+Total, Num is Numb/Total,
                (Value1=[]->Class=home; Class=ward).
```

Running results on probabilistic-rule induction. For the demonstration purpose, we show the final result of probabilistic rule induction with a specified minimum threshold 0.02. Each result has been formatted as probability >> rule's conditions (shown as attribute-value pairs) >> decision on class value (either home or ward).

```
0.0930233>>[comfort=10, bloodPressure=high, surfaceTemp=low]>>home
0.0581395>>[comfort=10, bloodPressure=mid, coreTempStability=stable,
            internalTemp=mid, bpStability=stable, surfaceTemp=mid,
            tempStability=unstable]>>home
0.0465116>>[comfort=10, bloodPressure=high, surfaceTemp=mid,
            bpStability=mod_stable]>>home
0.0348837>>[comfort=15, bpStability=unstable, surfaceTemp=mid]>>home
0.0348837>>[comfort=15, bpStability=stable, internalTemp=mid,
            tempStability=stable]>>home
0.0348837>>[comfort=10, bloodPressure=mid, coreTempStability=stable,
            internalTemp=mid, bpStability=unstable, surfaceTemp=mid,
            tempStability=stable]>>home
0.0348837>>[comfort=10, bloodPressure=low]>>home
0.0348837>>[comfort=10, bloodPressure=high, surfaceTemp=mid,
            bpStability=stable, oxygenSaturation=excellent]>>home
0.0232558>>[comfort=15, bpStability=unstable, surfaceTemp=high]>>home
0.0232558>>[comfort=15, bpStability=mod_stable]>>home
0.0232558>>[comfort=10, bloodPressure=mid, coreTempStability=unstable,
            tempStability=unstable, bpStability=stable]>>ward
0.0232558>>[comfort=10, bloodPressure=mid, coreTempStability=stable,
            internalTemp=mid, bpStability=stable, surfaceTemp=low]>>home
0.0232558>>[comfort=10, bloodPressure=mid, coreTempStability=stable,
            internalTemp=mid, bpStability=mod_stable,
            surfaceTemp=high]>>home
0.0232558>>[comfort=10, bloodPressure=mid, coreTempStability=stable,
            internalTemp=low, surfaceTemp=low,
            tempStability=stable]>>home
0.0232558>>[comfort=10, bloodPressure=mid, coreTempStability=stable,
            internalTemp=high]>>home
```

4. CONCLUSION

Modern healthcare organizations generate huge amount of electronic data stored in heterogeneous databases. Data collected by hospitals and clinics are not yet turned into useful knowledge due to the lack of efficient analysis tools. We thus propose a rapid prototyping of an automatic knowledge-mining tool to induce probabilistic knowledge from medical data. The induced knowledge is to be integrated into the knowledge base of a medical decision support system. Thus, in our design the knowledge base will be composed of precise knowledge as well as a minimal set of induced probabilistic knowledge. Discovered knowledge can also facilitate the reuse of knowledge base among decision-support applications within organizations that own heterogeneous clinical and health databases. Direct application of medical probabilistic knowledge base is for medical related decision-making. Other indirect but obvious application of such knowledge is to pre-process other data sets by grouping it into focused subset containing only relevant data instances.

The main contribution of this work is our implementation of knowledge mining engines based on the concept of higher-order Horn clauses using Prolog language. Higher-order programming has been originally appeared in functional languages in which functions can be passed as

arguments to other functions and can also be returned from other functions. This style of programming has soon been ubiquitous in several modern programming languages such as Perl, PHP, and JavaScript. Higher order style of programming has shown the outstanding benefits of code reuse and high level of abstraction. This paper illustrates higher order programming techniques in SWI-Prolog. The powerful feature of meta-level programming in Prolog facilitates the reuse of mining results represented as rules to be flexibly applied as conditional clauses in other applications.

The plausible extensions of our current work are to add constraints into the knowledge mining method in order to limit the search space and therefore yield the most relevant and timely knowledge, and due to the uniform representation of Prolog's statements as a clausal form, mining from the previously mined knowledge should be implemented naturally. The probabilistic knowledge induction and inferring techniques presented in this paper can be applied to the development of probabilistic databases. We also plan to extend our system to work with stream data that normally occur in modern medical organizations.

REFERENCES

1. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. University of California, Irvine, CA (2007) http://archive.ics.uci.edu/ml/
2. Bojarczuk, C.C. et al.: A Constrained-Syntax Genetic Programming System for Discovering Classification Rules: Application to Medical Data Sets. Artificial Intelligence in Medicine 30, 27--48 (2004)
3. He, Y., Tang, Y., Zhang, Y., Sunderraman, R.: Adaptive Fuzzy Association Rule Mining for Effective Decision Support in Biomedical Applications. Int. J. Data Mining and Bioinformatics 1, 1, 3--18 (2006)
4. Kononenko, I.: Machine Learning for Medical Diagnosis: History, State of the Art and Perspective. Artificial Intelligence in Medicine 1, 89--109 (2001)
5. Kretschmann, E. et al.: Automatic Rule Generation for Protein Annotation with the C4.5 Data Mining Algorithm Applied on SWISS-PROT. Bioinformatics 17, 10, 920--926 (2001)
6. Li, J., Fu, A., Fahey, P.: Efficient Discovery of Risk Patterns in Medical Data. Artificial Intelligence in Medicine (2008) doi:10.1016/j.artmed.2008.07.008
7. Mugambi, E. et al.: Polynomial-Fuzzy Decision Tree Structures for Classifying Medical Data. Knowledge-Based System 17, 2-4, 81--87 (2004)
8. Nadathur, G., Miller, D.: Higher-Order Horn Clauses. J. ACM 37, 777--814 (1990)
9. Ordonez, C.: Comparing Association Rules and Decision Trees for Disease Prediction. In: Proc. Int. Workshop on Healthcare Information and Knowledge Management, pp.17--24 (2006)
10. Ohsaki, M., Sato, Y., Yokoi, H., Yamaguchi, T.: A Rule Discovery Support System for Sequential Medical Data in the Case Study of a Chronic Hepatitis Dataset. In: Proc. ECML/PKDD-2003 Discovery Challenge Workshop, http://www.lisp.vse.cz/challenge/ecml pkdd2003
11. Quinlan, J.R.: Induction of Decision Trees. Machine Learning 1, 81--106 (1986)
12. Roddick, J.F. et al.: Exploratory Medical Knowledge Discovery: Experiences and Issues. ACM SIGKDD Explorations Newsletter 5, 1, 94--99 (2003)
13. Shillabeer, A., Roddick, J.F.: Establishing a Lineage for Medical Knowledge Discovery. In: Proc. 6th Australasian Conf. on Data Mining and Analytics, pp.29--37 (2007)
14. Zhou, Z., Jiang, Y.: Medical Diagnosis with C4.5 Rule Preceded by Artificial Neural Network Ensemble. IEEE Transactions on Information Technology in Biomedicine 1, 37--42 (2003)

ACKNOWLEDGMENTS

This research has been funded by grants from the National Research Council of Thailand (NRCT) and the second author is supported by the Thailand Research Fund (TRF, grant number RMU5080026). DEKD has been fully supported by Suranaree University of Technology.

Knowledge Mining with a Higher-Order Logic Approach

Kittisak Kerdprasop and Nittaya Kerdprasop

Abstract. Knowledge mining is the process of deriving new and useful knowledge from vast volumes of data and patterns previously discovered and stored as background knowledge. We propose a knowledge-mining system as a repertoire of tools for discovering strong and useful patterns. A pattern is strong if it represents frequently occurring relationships. Usefulness is achieved through constraints guided by users. To be able to derive strong and useful patterns from underlying data and background knowledge we consider employing the concept of higher-order logic as a major approach of our implementation. Higher-order logic can greatly reduce the burden of programmers as it is a very high level programming scheme suitable for the development of knowledge-intensive tasks. We have shown in this paper frequent pattern mining implemented with higher-order logic. The implementation is applied to mine breast cancer data. Our design of a logic-based knowledge-mining system is intended to support higher-order and constraint mining which is the next step of our research direction.

1 Introduction

Knowledge is a valuable asset to most organizations as a substantial source to support better decisions and thus to enhance organizational competency. Researchers and practitioners in the area of knowledge management view knowledge in a broad sense as a state of mind, an object, a process, an access to information, or a capability [2, 11]. The term *knowledge assets* [17, 19] is used to refer to any organizational intangible assets related to knowledge such as know-how, expertise, intellectual property. In clinical companies and computerized healthcare applications knowledge assets include order sets, drug-drug interaction rules, guidelines for practitioners, and clinical protocols [10].

Knowledge assets can be stored in data repositories either in implicit or explicit form. Explicit knowledge can be managed through the existing tools available in the current database technology. Implicit knowledge, on the contrary, is harder to

Kittisak Kerdprasop and Nittaya Kerdprasop
Data Engineering and Knowledge Discovery Research Unit,
School of Computer Engineering, Suranaree University of Technology,
Nakhon Ratchasima 30000, Thailand

K. Nakamatsu et al. (Eds.): New Advan. in Intel. Decision Techno., SCI 199, pp. 151–159.
springerlink.com

achieve and retrieve. Specific tools and suitable environments are needed to extract such knowledge.

Implicit knowledge acquisition can be achieved through the availability of the knowledge-mining system. *Knowledge mining* is the discovery of hidden knowledge stored possibly in various forms and places in large data repositories. In health and medical domains, knowledge has been discovered in different forms such as association rules, classification, clustering, trend or temporal pattern analysis [20]. The discovered knowledge facilitates expert decision support, diagnosis and prediction.

In this paper we present the design of a complete knowledge-mining system to support a high-level decision not only in medical domains but also in any domain that requires a knowledge-based decision support. A rapid prototyping of the proposed system is also provided to highlight the fact that higher-order logic is an appropriate approach to the implementation of a complex knowledge-mining system. The intuitive idea of our design and implementation is that for such a complicated knowledge-based system program coding should be done declaratively at a high level to alleviate the burden of programmers. The declarative style of programming also eases the future extension of our system to cover the concepts of higher-order mining [16] and constraint programming [6] that should naturally applied to the task of knowledge mining.

The rest of this paper is organized as follows. Section 2 reviews related works. Section 3 is the architecture of SUT-Miner, the proposed knowledge-mining system. Section 4 shows the implementation of association mining [1] using higher-order logic programming scheme with some running examples. Section 5 concludes the paper and discusses our future research directions.

2 Related Works

In recent years we have witnessed increasing number of applications devising database technology and machine learning techniques to mine knowledge from biomedicine, clinical and health data. Roddick et al [15] discussed the two categories of mining techniques applied over medical data: explanatory and exploratory. Explanatory mining refers to techniques that are used for the purpose of confirmation or making decisions. Exploratory mining is data investigation normally done at an early stage of data analysis in which an exact mining objective has not yet been set.

Explanatory mining in medical data has been extensively studied in the past decade employing various learning techniques. Bojarczuk et al [3] applied genetic programming method with constrained syntax to discover classification rules from medical data sets. Thongkam et al [21] studied breast cancer survivability using AdaBoost algorithms. Ghazavi and Liao [7] proposed the idea of fuzzy modeling on selected features medical data. Huang et al [9] introduced a system to apply mining techniques to discover rules from health examination data. Then they employed a case-based reasoning to support the chronic disease diagnosis and treatments. The recent work of Zhuang et al [25] also combined mining with case-based reasoning, but applied a different mining method. They performed data clustering based on self-organizing maps in order to facilitate decision support on solving new cases of pathology test ordering problem. Biomedical discovery

support systems are recently proposed by a number of researchers [4, 5, 8, 23, 24]. Some works [14, 18] extended medical databases to the level of data warehouses.

Exploratory, as oppose to explanatory, is rarely applied to medical domains. Among the rare cases, Nguyen and Kawasaki [13] introduced knowledge visualization in the study of hepatitis patients. Palaniappan and Ling [14] applied the functionality of OLAP tools to improve visualization.

It can be seen from the literature that most medical knowledge discovery systems have applied only some mining techniques such as classification rules mining, association mining, data clustering to discover hidden patterns and knowledge. We, on the contrary, design a knowledge-mining system aiming at providing a suite of tools to facilitate users and medical practitioners on discovering different kinds of knowledge from their data and background knowledge repositories.

3 SUT-Miner: A Knowledge-Mining system

Knowledge mining is a complex and possibly iterative process that involves many different steps. The main input to the process is data from heterogeneous sources, and the final output is the useful information desired by the users. For the medical domains, we design the system to be composed of two main phases as shown in figure 1. Knowledge induction phase is the back-end of the system responsible for acquiring and discovering new and useful knowledge. Usefulness is to be validated at the final step by human experts. Discovered knowledge is stored in the knowledge base to be applied to solve new cases in knowledge inferring phase which is the front-end of the proposed system.

SUT-Miner in a knowledge induction phase is comprised of three main modules: pre-DM, DM, post-DM. The term data mining (DM) means automatic learning of patterns or models from specific data. *Pattern* is an expression describing a subset of the data, *e.g.* $f(x) = 3x^2 + 3$ is a pattern induced from a given dataset {(0,3), (1,6), (2,15), (3,30)}, whereas the term *model* refers to a representation of the source generating the data, *e.g.* $f(x) = ax^2 + b$. In his paper we refer to both patterns and models as new knowledge discovered from data sources.

The pre-DM module performs data preparation tasks such as to locate and access relevant data sets, transform the data format, clean the data if there exists noise and missing values, reduce the data to a reasonable and sufficient size with only relevant attributes. The DM module performs mining tasks including classification, prediction, clustering, and association. We adopt the ontology concept at this step to guide the mining methodology selection. A simple form of mining method ontology is shown in figure 2. The post-DM module is composed of two main components: knowledge evaluator and knowledge integrator. These components perform major functionalities aiming at a feasible knowledge deployment which is important for the applications in medical diagnosis and predicting. Knowledge evaluator involves evaluation, based on corresponding measurement metrics, of the mining results. Knowledge integrator examines the induced patterns to remove redundant knowledge. Ontology has also been applied at this step to provide essential semantics regarding the domain problems.