AI & ML PROJECT

CHIDROOP

AP20110010154

CSE/C

Code:

```
menu(US,GH,P_list,N_list):-
write('This program will build a concept space model
from training examples and then
classify an unseen sample'),nl,
write('Enter a selection followed by a period.'),nl,
write('1. Enter a positive instance'),nl,
write('2. Enter a negative instance'),nl,
write('3. Show the concept hypothesis so far'),nl,
write('4.Enter a test sample'),nl,
write('5. Exit'),nl,
write(SH),nl,
write(GH),nl,
write(P),nl,
write(N),nl,
read(Choice),
run_option(Choice,SH,GH,P,N),nl,menu(US,GH,P_list,N_li
st).
run_option(X,SH,GH,P,N):- X==1,write('enter a positive
instance
'),read(P_instance),append(P,P_instance,P_list),
generalize(SH,P_instance,US),menu(US,GH,P_list,N),nl.
run_option(X,SH,GH,P,N):-X==2,write('enter a negative
instance
'),read(N_instance),append(N,N_instance,N_list),
specialize(GH,N_instance,US),menu(US,SH,N_list,P),nl.
run_option(X,SH,GH,P,N):-X==3,write('show the concept
```

```
of
hypothesis'),read(N_instance),append(N,N_instance,N_li
st),specialize(GH,N_instance,US),menu(US,SH,N_list,P),
nl.
run_option(X,SH,GH,P,N):-X==4,write('Test a
sample'),read(N_instance),append(N,N_instance,N_list),
specialize(GH,N_instance,US),menu(US,SH,N_list,P),nl.
run_option(X,SH,GH,P,N):-X==5,write('Exit').
run_candidate_elim :- candidate_elim([[_,_,_]], [],
[[small, medium, large], [red, blue, green], [ball,
brick, cube]]).
candidate_elim([G],[S],_) :-
covers(G,S),covers(S,G),
write("target concept is "), write(G),nl.
candidate_elim(G, S, Types) :-
write("G= "), write(G),nl,
write("S= "), write(S),nl,
write("Enter Instance "),
read(Instance),
process(Instance, G, S, Updated_G, Updated_S, Types),
candidate_elim(Updated_G, Updated_S, Types).
process(negative(Instance), G, S, Updated_G,
Updated_S, Types) :-
delete(X, S, covers(X, Instance), Updated_S),
specialize_set(G,Spec_G, Instance, Types),
delete(X, Spec_G, (member(Y, Spec_G), more_general(Y,
X)), Pruned_G),
delete(X, Pruned_G, (member(Y, Updated_S),
not(covers(X, Y))), Updated_G).
process(positive(Instance), G, [], Updated_G,
[Instance],_):-
delete(X, G, not(covers(X, Instance)), Updated_G).
process(positive(Instance), G, S, Updated_G,
Updated_S,_) :-
delete(X, G, not(covers(X, Instance)), Updated_G),
generalize_set(S,Gen_S, Instance),delete(X, Gen_S,
(member(Y, Gen_S), more_general(X, Y)), Pruned_S),
delete(X, Pruned_S, not((member(Y, Updated_G),
```

```
covers(Y, X))), Updated_S).
process(Input, G, P, G, P,_):-
Input \= positive(_),Input \=
negative(_),
write("Enter either positive(Instance) or
negative(Instance) "), nl.
specialize_set([], [], _, _).
specialize_set([Hypothesis|Rest],Updated_H,Instance,
Types):-
covers(Hypothesis, Instance),
(bagof(Hypothesis, specialize(Hypothesis, Instance,
Types), Updated_head);
Updated_head = []),
specialize_set(Rest, Updated_rest, Instance, Types),
append(Updated_head, Updated_rest, Updated_H).
specialize_set([Hypothesis|Rest],[Hypothesis|Updated_r
est],Instance, Types):-
not(covers(Hypothesis, Instance)),
specialize_set(Rest,Updated_rest, Instance, Types).
specialize([Prop|_], [Inst_prop|_],
[Instance_values|_]):-
var(Prop),
member(Prop, Instance_values),
Prop \= Inst_prop.
specialize([_|Tail], [_|Inst_tail], [_|Types]):-
specialize(Tail, Inst_tail, Types).
generalize_set([], [], _).
generalize_set([Hypothesis|Rest],Updated_H,Instance):-
not(covers(Hypothesis, Instance)),
(bagof(X, generalize(Hypothesis, Instance, X),
Updated_H); Updated_head =
[]),generalize_set(Rest,Updated_rest, Instance),
append(Updated_head, Updated_rest, Updated_H).
generalize_set([Hypothesis|Rest],[Hypothesis|Updated_r
est],Instance):-
covers(Hypothesis, Instance),
generalize_set(Rest,Updated_rest, Instance).
generalize([],[],[]).
```

```
generalize([Feature|Rest], [Inst_prop|Rest_inst],
[Feature|Rest_gen]) :-
not(Feature \= Inst_prop),
generalize(Rest, Rest_inst, Rest_gen).
generalize([Feature|Rest], [Inst_prop|Rest_inst],
[_|Rest_gen]) :-
Feature \= Inst_prop,
generalize(Rest, Rest_inst, Rest_gen).
more_general(X, Y) :- not(covers(Y, X)), covers(X, Y).
covers([],[]).
covers([H1|T1], [H2|T2]) :-
var(H1), var(H2),
covers(T1, T2).
covers([H1|T1], [H2|T2]) :-
var(H1), atom(H2),
covers(T1, T2).
covers([H1|T1], [H2|T2]) :-
atom(H1), atom(H2), H1 = H2,
covers(T1, T2).
delete(X, L, Goal, New_L) :-
(bagof(X, (member(X, L), not(Goal)), New_L);New_L =
[]).
```

Output:

```
?- menu(_, [_,_,_] , [] , []).
This program will build a concept space model from training examples and then
classify an unseen sample
Enter a selection followed by a period.
1. Enter a positive instance
2. Enter a negative instance
3. Show the concept hypothesis so far
4.Enter a test sample
5. Exit
```

1)Enter a positive instance

```
Enter Instance positive([small,red,ball]).
G= [[_8896,_8902,_8908]]
S= [[small,red,ball]]
```

2) Enter a negative instance

```
Enter Instance |: negative([large,green,cube]).
G= [[small,_10162,_10168],[_10108,red,_10120],[_10060,_10066,ball]]
S= [[small,red,ball]]
```

3) Show the concept hypothesis so far

```
Enter Instance |: negative([small,blue,brick]).
G= [[_10108,red,_10120],[_10060,_10066,ball]]
S= [[small,red,ball]]
Enter Instance |: positive([small,green,ball]).
G= [[_10060,_10066,ball]]
S= [[small,_12976,ball]]
```

4) Enter a test sample

```
Enter Instance |: positive([large,red,ball]).
target concept is [_10060,_10066,ball]
true .
```

5) Exit

```
% Break level 1
[1]  ?-
```