AI-Assignment-5

Abhinav Saurabh MT20127

- First I have made an interface.py to extract the facts and store them into the file which can be used further by the prolog program.
- Then prolog program takes the input from the facts saved in the text file from the previous program.
- The further program gives the recommended suggestions.

Interface Program

- I have used python programming and Natural language processing libraries to get the facts from the input sentences.
- Preprocessing steps.
    - Lowering the words
    - Removing Punctuations
    - Lemmatizing words
    - Removed stop words
- Keywords are extracted and stored in the facts.txt.
- Following format is followed
    - yes(‘keyword’)
    - no(‘keyword’)

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import string
import warnings
warnings.filterwarnings('ignore')

nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')

inplist = []
stopWords = set(stopwords.words('english'))

wordnet_lemmatizer = WordNetLemmatizer()

print("What are your interests and hobbies ?\n")  #Taking user input

inp1 = input()

text = inp1.lower()         #lowering the words
for sgn in string.punctuation:    #removing punctuations
  text= text.replace(sgn, ' ')


text = wordnet_lemmatizer.lemmatize(text)     #lemmatizing_words


tok1 = word_tokenize(text)      #extract tokens

for wod in tok1:
```

```
for wod in tok1:
    if wod not in stopWords:    #remove stop words
      inplist.append(wod)


f = open("facts.txt", 'w')

oracle = ['python','programming','ml','database','advanceprogramming','probability','discreetmaths','cn',

for key in oracle:
  if key in inplist:
      f.write("yes('")
      f.write(key)
      f.write("').\n")
  else:
      f.write("no('")
      f.write(key)
      f.write("').\n")

inp1 = input("Are you interested in Artificial Intelligence?")
f.write("ai(")
f.write(inp1)
f.write(").\n")

inp2 = input("Are you interested in Data Engineering?")
f.write("de(")
f.write(inp2)
f.write(").\n")

inp3 = input("Are you interested in Information Security?")
f.write("is(")
f.write(inp3)
f.write(").\n")

inp4 = input("Are you interested in Mobile Computing?")
f.write("mc(")
f.write(inp4)
f.write(").\n")



f.close()
```

Input given:

```
I am interested in ai and have some programming experience. I also know python.
Are you interested in Artificial Intelligence?yes
Are you interested in Data Engineering?no
Are you interested in Information Security?no
Are you interested in Mobile Computing?no
```

The output obtained in facts.txt.

```
yes('python').
yes('programming').
no('ml').
no('database').
no('advanceprogramming').
no('probability').
no('discreetmaths').
no('cn').
no('electronics').
no('cryptography').
ai(yes).
de(no).
is(no).
mc(no).
```

Prolog Program:

The program will give the output of the electives using the facts generated by the previous interface program.

```
get_advice:-


    advice(_),
    preferences(List),nl,

    (isempty(List)
        ->write('Sorry cannot recommend you anything' ),nl
        ;write('Electives for you are :'),show(List)
    ),
    retractall(recommend(_)),
    retractall(ai(_)),
    retractall(de(_)),
    retractall(is(_)),
    retractall(mc(_)),
    clear.

get_facts:-
  open('/Users/abhinavsaurabh/Desktop/IIITD/Abhinav-MT20127/facts.txt', read, Str),
  read_file(Str,Lines),
  close(Str),
  facts(Lines).

facts([Head|Tail]):-
        ((Head == end_of_file)
            ->facts(Tail)
            ;assert(Head), facts(Tail)
            ).
facts([]).

advice('Machine Learning') :- ml,fail.
advice('Information Retrieval') :- ir,fail.
advice('Data Mining') :- dm,fail.
advice('Mobile Computing') :- mc,fail.
advice('Collaborative Filtering') :- cf, fail.
advice('Big Data Mining in Healthcare') :- bdmh,fail.
advice('Artificial Intelligence') :- ai,fail.
advice('Deep learning') :- dl,fail.
advice('Applied Cryptography') :- ac,fail.
advice('Advanced Machine Learning') :- aml,fail.
advice('Big Data Analytics') :- bda,fail.
advice('Distributed Systems Security') :- dss,fail.
advice('Embedded Systems') :- es,fail.
```

```
advice('Cellular Data Networks') :- cdn,fail.
advice('Network Security') :- ns,fail.
advice('Ad Hoc Wireless Networks') :- ahwn,fail.
advice('Secure Coding') :- sc,fail.
advice('Sorry, No Recommendation !').


preferences([Head|Tail]):- retract(recommend(Head)), preferences(Tail).
preferences([]).

show([Head|Tail]):-
    format('~n ~w',[Head]),show(Tail).

show([]).

isempty([]).


ml :-
    retract(ai(A)),
    assert(ai(A)),
    retract(de(D)),
    assert(de(D)),
    ((A == yes ; D == yes)
        ->true
        ;fail
        ),
    questioninterest('python'),
    questioninterest('programming'),
    assert(recommend('Machine Learning')).

ir :-
    retract(ai(A)),
    assert(ai(A)),
    retract(de(D)),
    assert(de(D)),
    ((A == yes ; D == yes)
        ->true
        ;fail
        ),
    questioninterest('python'),
    questioninterest('programming'),
    questioninterest('database'),
```

```
    questioninterest('advancedprogramming'),
    assert(recommend('Information Retrieval')).

dm :-
    retract(ai(A)),
    assert(ai(A)),
    retract(de(D)),
    assert(de(D)),
    ((A == yes ; D == yes)
        ->true
        ;fail
        ),
    questioninterest('python'),
    questioninterest('programming'),
    questioninterest('probability'),
    assert(recommend('Data Mining')).


mc :-
        retract(mc(M)),
        assert(mc(M)),

        (M == yes
            ->true
            ;fail
            ),
        questioninterest('programming'),
        assert(recommend('Mobile Computing')).

cf :-
    retract(de(D)),
    assert(de(D)),
    ( D == yes
        ->true
        ;fail
        ),
    questioninterest('python'),
    questioninterest('programming'),
    questioninterest('ml'),
    assert(recommend('Collaborative Filtering')).

bdmh :-
    retract(ai(A)),
```

```
    retract(ai(A)),
    assert(ai(A)),
    retract(de(D)),
    assert(de(D)),
    ((A == yes ; D == yes)
        ->true
        ;fail
        ),

    questioninterest('ml'),
    assert(recommend('Big Data in Healthcare')).

ai :-
    retract(ai(A)),
    assert(ai(A)),
    (A == yes
        ->true
        ;fail
        ),

    assert(recommend('Artificial Intelligence')).




dl :-
    retract(ai(A)),
    assert(ai(A)),
    (A == yes
        ->true
        ;fail
        ),

    questioninterest('ml'),
    assert(recommend('Deep Learning')).


ac :-
    retract(is(A)),
    assert(is(A)),
    (A == yes
        ->true
        ;fail
```

```
        ),
    questioninterest('discreetmaths'),
    assert(recommend('Applied Cryptography')).

aml :-
    retract(ai(A)),
    assert(ai(A)),

    (A == yes
        ->true
        ;fail
        ),

    questioninterest('ml'),
    assert(recommend('Advanced Machine Learning')).

bda :-
    retract(de(D)),
    assert(de(D)),
    ( D == yes
        ->true
        ;fail
        ),

        questioninterest('programming'),
        questioninterest('database'),
        assert(recommend('Big Data Analytics')).

dss :-
    retract(is(A)),
    assert(is(A)),
    retract(mc(D)),
    assert(mc(D)),
    ((A == yes ; D == yes)
        ->true
        ;fail
        ),
  questioninterest('cn'),
    assert(recommend('Distributed System Security')).

es :-
        retract(mc(D)),
        assert(mc(D)),
```

```
214         assert(mc(D)),
215         (D == yes
216             ->true
217             ;fail
218             ),
219         questioninterest('electronics'),
220         assert(recommend('Embedded Systems')).
221
222 cdn :-
223             retract(mc(D)),
224             assert(mc(D)),
225             ( D == yes
226                 ->true
227                 ;fail
228                 ),
229           questioninterest('cn'),
230             assert(recommend('Cellular Data Networks')).
231
232 ns :-
233                 retract(is(A)),
234                 assert(is(A)),
235                 retract(mc(D)),
236                 assert(mc(D)),
237                 ((A == yes ; D == yes)
238                     ->true
239                     ;fail
240                     ),
241               questioninterest('cn'),
242                 assert(recommend('Network Security')).
243
244 ahwn :-
245                     retract(mc(D)),
246                     assert(mc(D)),
247                     (D == yes
248                         ->true
249                         ;fail
250                         ),
251                   questioninterest('cn'),
252                     assert(recommend('Ad Hoc Wireless Networks')).
253
254
255 sc :-
256                         retract(is(A)),
```

```
                        assert(is(A)),

                        (A == yes
                            ->true
                            ;fail
                            ),
                      questioninterest('cryptography'),
                        assert(recommend('Secure Coding')).


questioninterest(In) :-
    (yes(In)
        ->true
        ;(no(In)
            ->fail
            ;ask(In))
    ).

ask(Que) :-
    format('~w ?',[Que]),
    read(Ans),
    ( (Ans == yes ; Ans == y)
        ->assert(yes(Que))
        ;assert(no(Que)), fail
        ).

:- dynamic yes/1,no/1.


clear :- retract(yes(_)),fail.
clear :- retract(no(_)),fail.
clear.


read_file(Stream,[]) :-
  at_end_of_stream(Stream).
read_file(Stream,[X|L]) :-
  \+ at_end_of_stream(Stream),
  read(Stream,X),
  read_file(Stream,L).
```

Output:

```
(base) abhinavsaurabh@Abhinavs-MacBook-Pro ~ % swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['/Users/abhinavsaurabh/Desktop/IIITD/Abhinav-MT20127/a5.pl'].
true.

?- get_facts.
true .

?- get_advice.

Electives for you are :
 Machine Learning
 Artificial Intelligence
true
```