Assignment 5

Pipeline :-

There are 2 files to be run → interface.py and prolog_code.pl .
interface.py is the python interface and prolog_code.pl is the electives advisory system in prolog.
First, we have to run interface.py, which will extract the facts(using nlp techniques) from the inputs the student gives, and store it in a temporary file (facts.pl).
Then, we will run prolog_code.pl which will take the inputs from the temporary file as facts, and will then give the recommendation of the electives the student can take.

Assumptions :-

Some assumptions made are that the interface will not take multiple departments or grading criteria in input. It also won't take inputs where the adverb "not" is associated, such as "I am not from the cse department".

Source code with comments of python interface file :-

(Comments are written in green color and start with #)

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from string import punctuation

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

# creating set of stop words
stop_words = set(stopwords.words("english"))

# using a word lemmatizer to group together different types of similar
word
lemmatizer = WordNetLemmatizer()
```

```python
# this converts a text to lower case
def get_lower_case(text):
  return text.lower()

# this replaces the punctuations in a text with a space
def get_non_punctuated_words(text):
  output = ""
  for ch in text:
    if ch in punctuation:
      output += ' '
    else:
      output += ch
  return output


# this generates a tokenized list from a text
def get_tokenized_list(text):
  return word_tokenize(text)


# this removes the stopwords present in the list arr
def get_non_stopword_list(arr):
  # arr is tokenized already
  l = []
  for word in arr:
    if word not in stop_words:
      l.append(word)
  return l

# this lemmatizes the words present in the list arr
def get_lemmatized_wordlist(arr):
  # arr is tokenized already
  l = []
  for word in arr:
    l.append(lemmatizer.lemmatize(word))
  return l


f = open("facts.pl", "w")
```

```
# read input for the branch/department of student, then preprocess it
inp1 = input("What is your branch/department ?\n: ")
text = get_lower_case(inp1)
text = get_non_punctuated_words(text)
l = get_tokenized_list(text)
l = get_non_stopword_list(l)
l = get_lemmatized_wordlist(l)


# write down the fact of department/1 in the prolog file facts.pl
if 'cse' in l:
  f.write("department('CSE').\n")
else:
  f.write("department('Non-CSE').\n")


# read the input for all the interest areas of the student, then
preprocess it
inp2 = input("What are your interest areas ?\n: ")
text = get_lower_case(inp2)
text = get_non_punctuated_words(text)
l = get_tokenized_list(text)
l = get_non_stopword_list(l)
l = get_lemmatized_wordlist(l)


# write down the fact of interest/1 in the prolog file facts.pl
if ('machine' in l) or ('learning' in l) or ('ml' in l) or ('ai' in l) or
('artificial' in l) or ('intelligence' in l):
  f.write("interest('Machine Learning').\n")
if ('networking' in l) or ('network' in l) or ('security' in l):
  f.write("interest('Networking').\n")
if ('database' in l) or ('dbms' in l) or ('information' in l):
  f.write("interest('Databases').\n")
if ('algorithm' in l) or ('structure' in l) or ('dsa' in l) or ('coding'
in l):
  f.write("interest('Algorithms').\n")
if ('biology' in l) or ('gastronomy' in l) or ('medicine' in l) or
('doctor' in l):
  f.write("interest('Biology').\n")
if ('designing' in l) or ('animation' in l) or ('graphics' in l):
  f.write("interest('Design').\n")
if ('social' in l) or ('politics' in l) or ('ssh' in l) or ('ev' in l) or
('environmental' in l) or ('sociology' in l):
  f.write("interest('Social Science').\n")
if ('electronics' in l) or ('signal' in l) or ('digital' in l) or
('circuit' in l):
  f.write("interest('Electronics').\n")


# read the input for the grading policy the student prefers, then
preprocess it
inp3 = input("Do you prefer relative grading or absolute grading ?\n: ")
text = get_lower_case(inp3)
text = get_non_punctuated_words(text)
l = get_tokenized_list(text)
l = get_non_stopword_list(l)
l = get_lemmatized_wordlist(l)


# write down the fact of relative/1 in prolog file facts.pl
if 'relative' in l:
  f.write("grading('relative').\n")
else:
  f.write("grading('absolute').\n")


# read all the courses the student has done previously, then preprocess it
all_prereqs = ['cse201', 'cse222', 'ece250', 'cse232', 'cse121', 'cse102',
'cse231', 'mth100', 'cse202', 'mth201', 'cse101']
inp4 = input("Enter the course-codes of the courses you have done\n: ")
text = get_lower_case(inp4)
text = get_non_punctuated_words(text)
l = get_tokenized_list(text)
l = get_non_stopword_list(l)
l = get_lemmatized_wordlist(l)

arr = []
for word in l:
  if word in all_prereqs:
    arr.append(word.upper())

# write down the fact of courses_done/1 in prolog file facts.pl
f.write('courses_done('+str(arr)+').')
f.close()
```

Explanation of code:-

This is the python program for the natural language interface to provide inputs to the electives advisory system(prolog code). We have used the nltk library for natural language processing tasks. The main aim is to store the department, interest, grading criteria and courses done by the student as facts in a prolog file(facts.pl), so that we can read and load those facts in our electives advisory system.

First, we ask the student about his department/branch. We read the input as a sentence. Then we preprocess it by the following steps :-

- Convert the input to lowercase
- Replace the punctuations with space
- Tokenize the sentence
- Remove the stopwords from the tokenized list
- Lemmatize the list further to group specific words into a common one

After these preprocessing steps, we search for specific keywords such as “cse”. If it is present, then we store the fact **department(‘CSE’)**, otherwise we store the fact **department(‘Non-CSE’)**.
Similarly, we do the same process for other facts (interest/1, grading/1, courses_done/1) and search for specific keywords, which if present, we store the fact in facts.pl file.

Source code with comments of prolog advisory system file :-

(Comments are written in green letters and start with %)

```
% course fact contains --> Name of course, Course department, Area of
interest of course, Absolute/Relative grading, pre_reqs.

% Here we define which facts are dynamic
:- dynamic department/1, interest/1, grading/1, pre_reqs/1.
```

```
% Machine Learning Courses(CSE courses)
course('Machine Learning','CSE','Machine
Learning','relative',['MTH100','CSE101']).
course('Natural Language Processing','CSE','Machine
Learning','absolute',['MTH201','CSE101']).
course('Artificial Intelligence','CSE','Machine
Learning','relative',['CSE102','CSE121']).

% Networking Courses(CSE courses)
course('Computer
Networks','CSE','Networking','relative',['CSE101','CSE222','CSE232']).
course('Network
Security','CSE','Networking','absolute',['CSE232','CSE231']).
course('Computer Security','CSE','Networking','relative',['CSE232']).

% Databases Courses(CSE courses)
course('Information
Retrieval','CSE','Databases','relative',['CSE102','CSE201','CSE202']).
course('DBMS','CSE','Databases','relative',['CSE102']).
course('Information Integration and
Applications','CSE','Databases','absolute',['CSE202']).

% Algorithms Courses(CSE courses)
course('Algorithm Design and
Analysis','CSE','Algorithms','relative',['CSE102']).
course('Introduction to Programming','CSE','Algorithms','relative',[]).
course('Modern Algorithm
Design','CSE','Algorithms','absolute',['CSE222']).
course('Approximation
Algorithms','CSE','Algorithms','relative',['CSE222']).

% Biology Courses(Non-CSE courses)
course('Computational Gastronomy','Non-CSE','Biology','relative',[]).
course('Computing for Medicine','Non-CSE','Biology','relative',[]).
course('Machine Learning for Biomedical
Applications','Non-CSE','Biology','relative',[]).

% Design Courses(Non-CSE courses)
```

```
course('Introduction to Animation and
Graphics','Non-CSE','Design','absolute',[]).
course('3D Animation Film Making','Non-CSE','Design','relative',[]).
course('Affective
Computing','Non-CSE','Design','relative',['CSE101','CSE102','CSE201']).

% Social Science Courses(Non-CSE courses)
course('Neuroscience of Decision Making','Non-CSE','Social
Science','absolute',[]).
course('Urban Space and Political Power','Non-CSE','Social
Science','relative',[]).
course('Environmental Sciences','Non-CSE','Social
Sciences','relative',[]).

% Electronics Courses(Non-CSE courses)
course('Digital Signal
Processing','Non-CSE','Electronics','absolute',['ECE250']).
course('Reinforcement
Learning','Non-CSE','Electronics','relative',['MTH201']).
course('Digital Image
Processing','Non-CSE','Electronics','relative',['MTH100','MTH201']).


pre_req_satisfy([],L) :-
  is_list(L).
pre_req_satisfy(L,[]) :-
  (not(is_list(L)) -> false;
    L==[]).
pre_req_satisfy(L1,L2) :-
  % L1 is the pre-requisites of the given course
  % L2 is all the courses done by the student previously.
  % This function tells us if the student can take the given course on
basis of the
  % pre-requisites(L1) of given course and the courses the student has
done yet (L2).

  [H|T] = L1,
  member(H,L2),
  pre_req_satisfy(T,L2).
```

```
display_recommended_courses(A):-
  % This function is used to print all the courses that are best suitable
for the student based on his preferences taken.
  % The ranking of the courses are done based on --> department > interest
> grading > pre_req_satisfy.

  forall((course(Name,Dept,Area,Grading,Pre_reqs), department(Dept),
interest(Area), grading(Grading), pre_req_satisfy(Pre_reqs,A)),
(write(Name),nl)).



main :-
  % consult("facts.pl") loads the facts inside the file "facts.pl" in our
current prolog database.
  consult("facts.pl"),
  write('This is an Elective course recommendation system'),nl,

  write('What is your name?'),nl,
  read(StudName),nl,
  write('Hi '),write(StudName),nl,

  % Here we read the list of courses done previously by the student into
the variable A4.
  courses_done(A4),nl,

  % Here, we run the recommendation function
display_recommended_courses/1. This function
  % simply prints all the courses that the student can take which are
matching with the preferences
  % the student has selected before.
  write("Below are the recommended courses for you according to your given
preferences :-"),nl,nl,
  display_recommended_courses(A4).
```

<u>Explanation of code</u> :-

This is the prolog electives advisory system. Here we read the facts (department/1, interest/1, grading/1, courses_done/1) from a temporary file called facts.pl (it gets created every time the python code is run) .

The fact **department/1** specifies what type of courses the student prefers, CSE or Non-CSE. It has the value ‘CSE’ or ‘Non-CSE’ . If it is ‘CSE’, then the student can only take courses that have its department as ‘CSE’. Similarly, for ‘Non-CSE’.

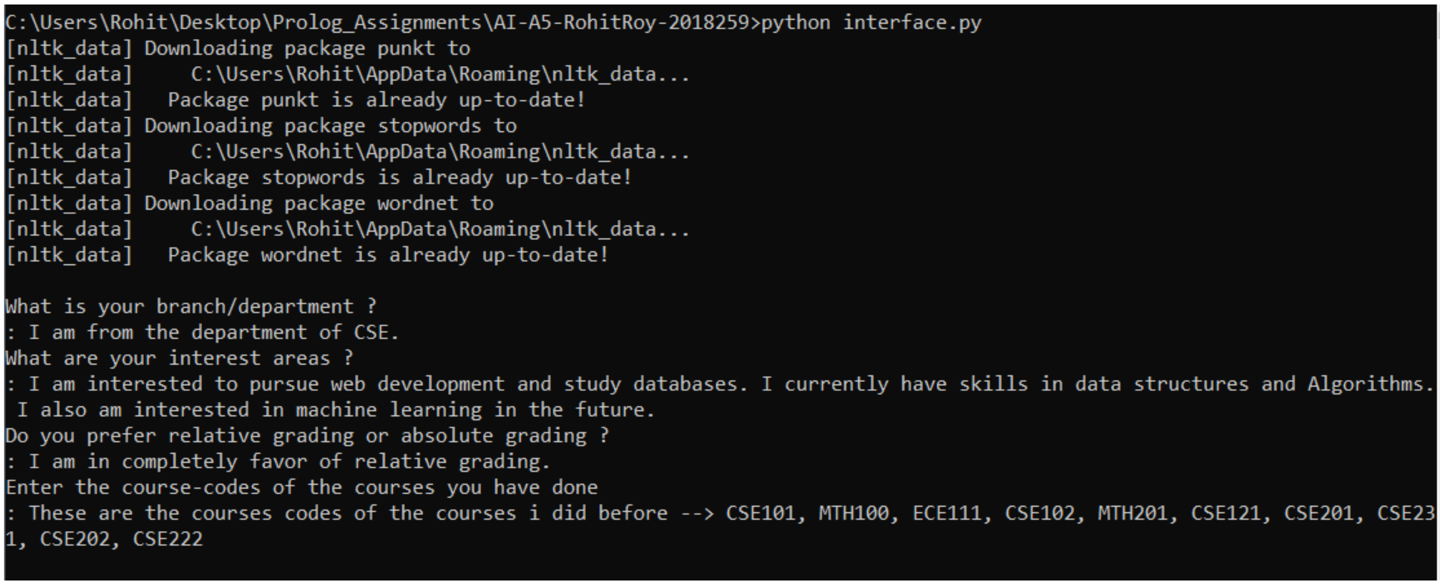
The fact **grading/1** specifies whether the student prefers relative or absolute grading courses. It has the value ‘relative’ or ‘absolute’. If it is ‘relative’, then the student can only take courses that have its grading criteria as relative. Similarly for ‘absolute’.

The fact **interest/1** specifies the interest areas of the student. It can have multiple values such as ‘Networking’, ‘Databases’, etc. There can be multiple **interest/1** facts, each one corresponding to a particular interest of the student. If the student has ‘Databases’ as one interest, then he can take courses of Databases (provided his other details match with the course).

The fact **courses_done/1** contains a list of course codes of courses done by the student previously. It is used to check whether the student has finished the pre-requisites of a course that he wants to take.

In this code, first we load the facts from the file facts.pl using the command **consult("facts.pl")**.
After all the facts have been loaded in the memory, we simply run the function (display_recommended_courses/1) which recommends/prints the courses that are preferable for the student. It takes an input of a list of course codes of the courses the student has done.

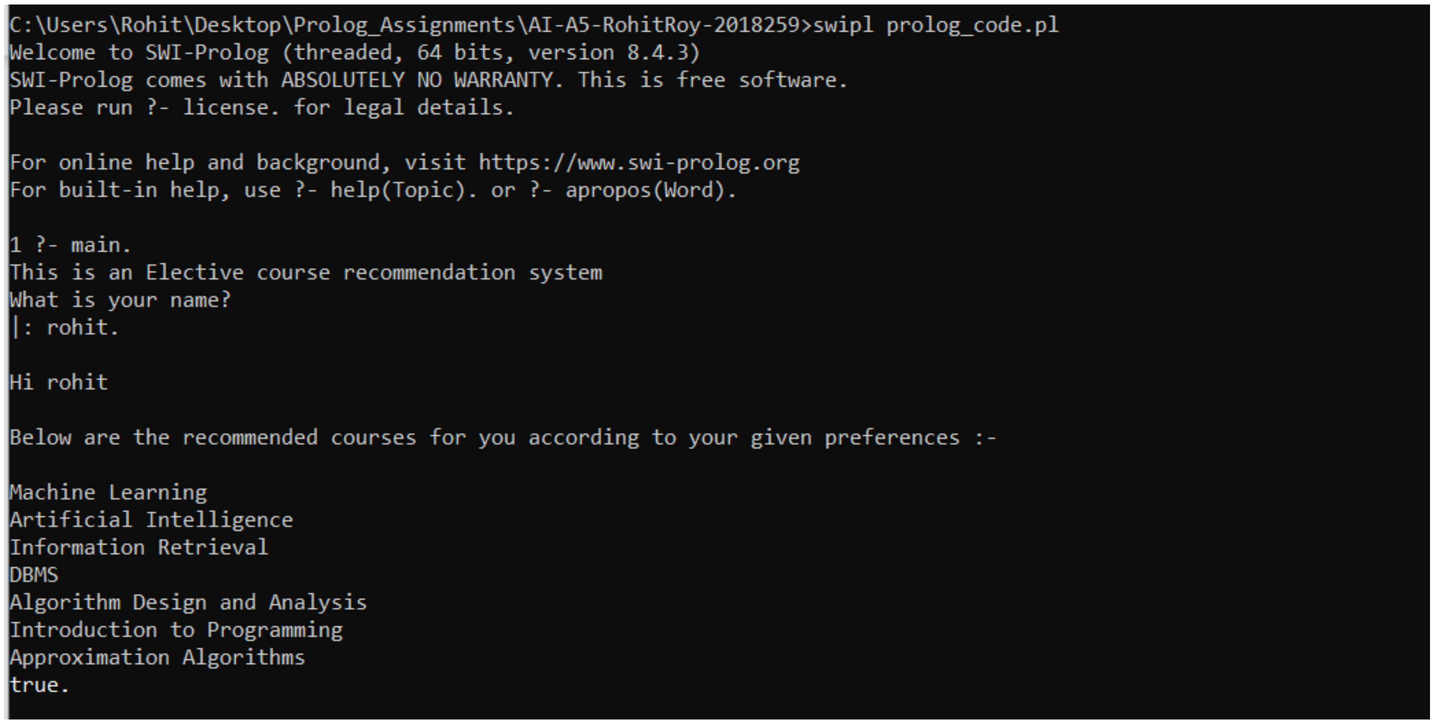
<u>Working Screenshot of python code</u>:-



```
C:\Users\Rohit\Desktop\Prolog_Assignments\AI-A5-RohitRoy-2018259>python interface.py
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Rohit\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Rohit\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Rohit\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!

What is your branch/department ?
: I am from the department of CSE.
What are your interest areas ?
: I am interested to pursue web development and study databases. I currently have skills in data structures and Algorithms.
 I also am interested in machine learning in the future.
Do you prefer relative grading or absolute grading ?
: I am in completely favor of relative grading.
Enter the course-codes of the courses you have done
: These are the courses codes of the courses i did before --> CSE101, MTH100, ECE111, CSE102, MTH201, CSE121, CSE201, CSE23
1, CSE202, CSE222
```

<u>Working Screenshot of prolog code</u>:-



```
C:\Users\Rohit\Desktop\Prolog_Assignments\AI-A5-RohitRoy-2018259>swipl prolog_code.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.3)
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).

1 ?- main.
This is an Elective course recommendation system
What is your name?
|: rohit.

Hi rohit

Below are the recommended courses for you according to your given preferences :-

Machine Learning
Artificial Intelligence
Information Retrieval
DBMS
Algorithm Design and Analysis
Introduction to Programming
Approximation Algorithms
true.
```