AI Assignment 5

Problem Statement:
You have to build a small natural language interface in Python that will provide inputs to your electives advisory system developed in Prolog.

Workflow:

1. A python program is developed to process the natural language data to advice electives to the user.
2. career_paths[] stores the different career paths used to decide electives in the prolog program as facts.
3. This data is stemmed to get root words.
4. The user is asked to input data about his interests, projects and stream.
5. This data is first tokenized and then stemmed.
6. Tokenized data is processed to store root words derived.
7. Check for stream and projects in tokens and assert root words accordingly.
8. Stemmed data is compared with stemmed career paths[] to check for common root words and the corresponding career paths are appended.
9. This roots[] list is further processed and facts are stored in facts.pl file.
10. This file is consulted by ‘main.pl’ which contains the ruleset for the electives advisory system.
11. The ideal electives are showcased to the user.

Output:

1. Preprocessing of NL data by python(using nltk library)

```
C:\Users\dell\Dropbox\My PC (DESKTOP-DKG8LHI)\Documents\Prolog\Sample_codes>python AI-A5-Arjun-2020497.py
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\dell\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!

Enter information about your interests, stream and projects:
I am in the cse stream. I like ml, sociology and psychology. I have done many course projects.

Stemmed career paths:
 ['secur', 'algorithm', 'comput', 'math', 'statist', 'ml', 'design', 'biolog', 'anthropolog', 'psycholog', 'econom', 'ai', 'so
ciolog', 'project']

Tokenised input data:
 ['I', 'am', 'in', 'the', 'cse', 'stream', '.', 'I', 'like', 'ml', ',', 'sociology', 'and', 'psychology', '.', 'I', 'have', 'd
one', 'many', 'course', 'projects', '.']

Stemmed tokens:
 ['i', 'am', 'in', 'the', 'cse', 'stream', '.', 'i', 'like', 'ml', ',', 'sociolog', 'and', 'psycholog', '.', 'i', 'have', 'don
e', 'mani', 'cours', 'project', '.', 'algorithm']

Filtered root words:
 ['ml', 'sociology', 'algorithms', 'psychology', 'projects']
```

2. Facts stored in the ‘facts.pl’ file

facts.pl

```
interest_in(ml).
interest_in(sociology).
interest_in(algorithms).
interest_in(psychology).
done(projects).
```

3. Output in prolog ‘main.pl’.

```
% c:/Users/dell/Dropbox/My PC (DESKTOP-DKG8LHI)/Documents/Prolog/Sample_codes/main.pl compiled 0.00 sec, 46 clauses
?- main.
##################################################
----------------------Welcome-----------------------
##################################################
     Electives Prediction System for IIIT Delhi

You are all set to go for your ideal elective now!!

Based on your inputs, We have finalised the below courses for you
COURSE CODE      COURSE TITLE
##################################################
soc222      Technology and Future of Work

psy222      Neuroscience of Decision Making

psy233      Cognition of Motor Movement

cse511      Introduction to Graduate Algorithms

cse512      Modern Algorithm Design

cse533      Data Mining

cse534      Machine Learning

cse536      Advanced Machine Learning

##################################################
See you again,
true
```

Conclusions:

- We have used projects, stream and career interests of the student to decide the electives. The electives are chosen from the ruleset created for assignment 1.
- We assume that the NL data contains no negative statements.
- The data is given in the form of simple affirmative statements.

Source code(.py):

```
#import libraries
import nltk
nltk.download('punkt')
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from pyswip import Prolog

#career paths used to determine electives in prolog file(used as facts)
career_paths = ['security', 'algorithms','computers', 'maths','statistics',
'ml', 'design',
          'biology', 'anthropology', 'psychology',
'economics','ai','sociology','projects']

#store the stemmed career paths
stemmed_facts=[]
#store the stemmed tokens of the input text
stemmed_tokens=[]

#function to read the input file. returns the text
def read_file(path):
    readFile= open(path, 'r')
    text = readFile.read()
    readFile.close()
    return text

#implements stemming on the tokens
def stemming(career_paths,stemmed_facts):
  ps = PorterStemmer()
  for c in career_paths:
    stem = ps.stem(c)
    stemmed_facts.append(stem)

#tokenises the input data in 'input.txt'
def tokenise(data):
    token=word_tokenize(data)
    print("\nTokenised input data: \n",token)
```

```
    return token

#checks for common stems in career_paths and appends the career_paths to roots
for annotation
def check_common_stems(stemmed_facts, stemmed_tokens):
  roots=[]
  for stem in stemmed_tokens:
    for i in range(len(career_paths)):
        if(stem == stemmed_facts[i]):
            roots.append(career_paths[i])
  return roots

#store tokens related to the branches
def check_for_branch(tokens):

 if('stream' in tokens):
    if 'csb' in tokens:
        tokens.append('biology')
    elif 'csd' in tokens:
        tokens.append('design')
    elif 'csai' in tokens:
        tokens.append('ai')
    elif 'cse' in tokens:
        tokens.append('algorithms')
    elif 'csss' in tokens:
        tokens.append('sociology')
        tokens.append('economics')
    elif 'csam' in tokens:
        tokens.append('math')

# handle special cases (processing multiple keywords in input data)
def complex_cases(tokens):
  if('theoretical' in tokens):
    if('computer' in tokens):
      tokens.append("computers")
    if('maths' in tokens or 'math' in tokens):
      tokens.append("statistics")

##########################################
#start here
##########################################
```

```
#retract all previous facts
f=open('facts.pl','w')
f.write("")
f.close()
text=input("\nEnter information about your interests, stream and projects: \n")
stemming(career_paths, stemmed_facts)
print("\nStemmed career paths: \n",stemmed_facts)
# text=read_file('input.txt')
tokens=tokenise(text)
check_for_branch(tokens)
complex_cases(tokens)
stemming(tokens, stemmed_tokens)
print("\nStemmed tokens: \n",stemmed_tokens)
roots=check_common_stems(stemmed_facts, stemmed_tokens)

#duplicate handling
roots = list(set(roots))
print("\nFiltered root words: \n", roots)



#storing facts in facts.pl
for i in range(len(roots)):
  if(roots[i] == 'projects'):
    tagstring="done("+roots[i]+").\n"
  else:
    tagstring="interest_in("+roots[i]+").\n"
  f = open('facts.pl', 'a')
  f.write(tagstring)

f.close()
```