Assignment 5

Mohammad Sufyan Azam, 2020312

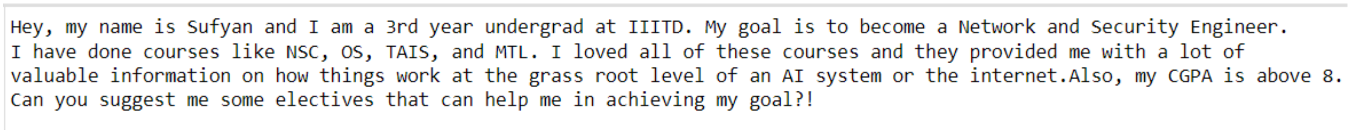
NLI interface for an Elective Advisory System

Steps followed -

- Imported all the required modules.
- Downloaded stopword, punctuations, wordnet and omw-1.4 for NLP preprocessing.
- Read the input file and stored it in a variable *text*. Then removed the punctuations and lemmatized them.
- Then tokenized the *text* variable and finally removed the stopWords.
- Defined the knowledge base and some helper functions for the program.
- Extracted the career interests and courses done using the helper functions.
- Wrote interest, and course done facts into a text file and stored the complete knowledge base too in that text file.
- Read that text file from a prolog program and dynamically asserted them into the knowledge base.
- Processed the given information to find out the best suitable electives for the user and printed them.

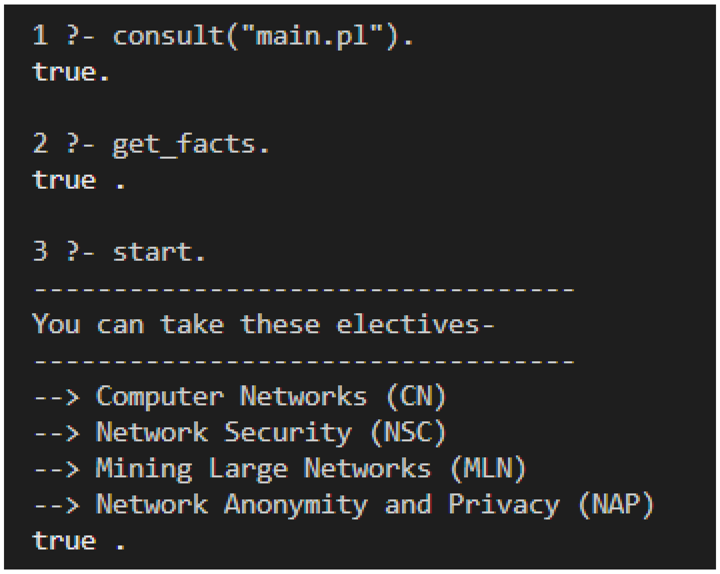
Screenshots of the program -

Input Text 1



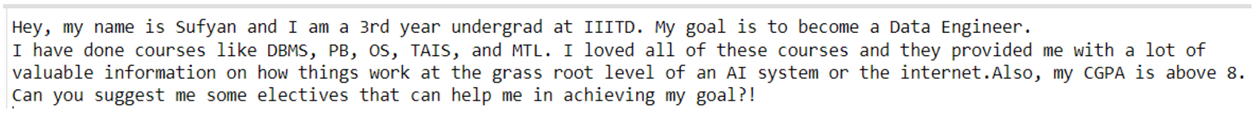
```
Hey, my name is Sufyan and I am a 3rd year undergrad at IIITD. My goal is to become a Network and Security Engineer.
I have done courses like NSC, OS, TAIS, and MTL. I loved all of these courses and they provided me with a lot of
valuable information on how things work at the grass root level of an AI system or the internet.Also, my CGPA is above 8.
Can you suggest me some electives that can help me in achieving my goal?!
```

Output of Prolog File



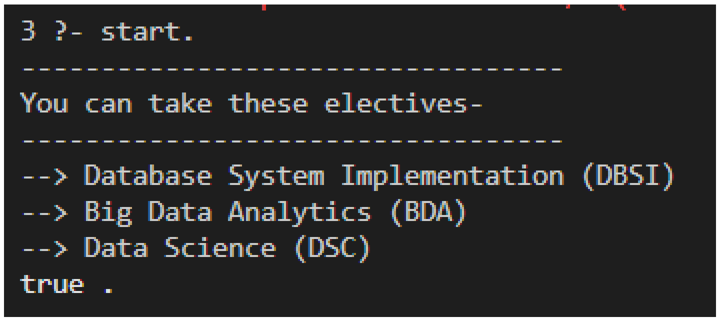
```
1 ?- consult("main.pl").
true.

2 ?- get_facts.
true .

3 ?- start.
----------------------------------
You can take these electives-
----------------------------------
--> Computer Networks (CN)
--> Network Security (NSC)
--> Mining Large Networks (MLN)
--> Network Anonymity and Privacy (NAP)
true .
```

Input Text 2



```
Hey, my name is Sufyan and I am a 3rd year undergrad at IIITD. My goal is to become a Data Engineer.
I have done courses like DBMS, PB, OS, TAIS, and MTL. I loved all of these courses and they provided me with a lot of
valuable information on how things work at the grass root level of an AI system or the internet.Also, my CGPA is above 8.
Can you suggest me some electives that can help me in achieving my goal?!
```

Output of Prolog File



```
3 ?- start.
----------------------------------
You can take these electives-
----------------------------------
--> Database System Implementation (DBSI)
--> Big Data Analytics (BDA)
--> Data Science (DSC)
true .
```

Code -

```
#!/usr/bin/env python
# coding: utf-8

# ## Importing All Essential Libraries

# In[1]:


import nltk
import numpy as np
from nltk.stem import WordNetLemmatizer
import pandas as pd
from nltk.corpus import stopwords
import sklearn
from nltk.tokenize import word_tokenize
import string
import warnings
from nltk.stem import PorterStemmer


# ## Downloading stopwords, punctuations, and wordnet

# In[2]:


nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')


# ## Reading Input File

# In[3]:


# Rules-
# Only one interest should be mentioned ideally.
# No spelling mistakes. Writes exact career and interest as defined in the
list.
# Write 'courses done' or 'done courses' preceeding the courses done by
you.

input_file = open("D:\My Folder\Academics\AI\AI_Assignment\AI Assignment
5\input_file.txt", 'r')
text = input_file.read()
input_file.close()


# ## Preprocessing Input File Using NLP Libraries

# In[4]:


input_words = []

# stop words from English
stopWords = set(stopwords.words('english'))

# Declaring the wordNet Lemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

text = text.lower()
# Removing the punctuations
for punctuation in string.punctuation:
  text = text.replace(punctuation, ' ')

# lemmatising the words and updating the variable text
text = wordnet_lemmatizer.lemmatize(text)

# tokenising text
tokenised_text = word_tokenize(text)

# removing the stop words here
for word in tokenised_text:
    if word not in stopWords:
      input_words.append(word)
```

```
print(input_words)


# ## Defining Courses and Interests

# In[5]:


interest_choices = ['Network and Security Engineer', 'Data Engineer',
'Electronics Engineer', 'Bioinformatics Engineer',
                    'Robotics Engineer', 'AI Engineer', 'ML Engineer']

data_engineer_courses = ['Database Management Systems (DBMS)', 'Database
System Implementation (DBSI)', 'Big Data Analytics (BDA)', 'Data Science
(DSC)']
ai_engineer_courses = ['Artificial Intelligence (AI)', 'Meta-Learning
(MTL)', 'Trustworthy AI Systems (TAIS)']
ml_engineer_courses = ['Statistical Machine Learning (SML)', 'Advanced
Machine Learning (AML)', 'Machine Learning (ML)',
                       'Natural Language Processing (NLP)']
robotics_engineer_courses = ['Robotics (IRob)', 'Social Robotics (SR)',
'Non Linear and Adaptive Control of Robotic Systems (NLR)']
electronics_engineer_courses = ['Integrated Electronics (IE)', 'Circuit
theory and devices (CTD)', 'Fields and Waves (F&W)',
                                'Embedded Logic Design (ELD)', 'Digital
Signal Processing (DSP)']
bioinformatics_engineer_courses = ['Practical Bioinformatics (PB)',
'Algorithms in BioInformatics (ABIN)',
'Algorithms in Computational Biology (ACB)', 'Computing For Medicine
(CM)', 'Computer Aided Drug Design (CADD)']
network_engineer_courses = ['Computer Networks (CN)', 'Network Security
(NSC)', 'Operating Systems (OS)', 'Mining Large Networks (MLN)', 'Network
Anonymity and Privacy (NAP)']
ai_engineer_courses += ml_engineer_courses

list_of_courses = network_engineer_courses + data_engineer_courses +
ai_engineer_courses + robotics_engineer_courses +
electronics_engineer_courses + bioinformatics_engineer_courses
# print(len(list_of_courses), list_of_courses)
```

```python
# ## Defining Helper Functions

# In[6]:


def get_index(word, listt):
    try:
        index = listt.index(word)
        return index
    except:
        return -1


def find_career_interest(index, listt):
    try:
        i = index - 1
        if listt[i] == 'security' and listt[i-1] == 'network':
            return interest_choices[0]
        elif listt[i] == 'data':
            return interest_choices[1]
        elif listt[i] == 'electronics':
            return interest_choices[2]
        elif listt[i] == 'bioinformatics':
            return interest_choices[3]
        elif listt[i] == 'robotics':
            return interest_choices[4]
        elif listt[i] == 'ai':
            return interest_choices[5]
        elif listt[i] == 'ml':
            return interest_choices[6]
    except:
        raise ValueError


def find_courses_done(index, listt):
    try:
        if listt[index-1] != 'done' and listt[index+1] != 'done':
            return []
        i = index
        if listt[i-1] == 'done':
            i = i+1
        elif listt[i+1] == 'done':
            i = i+2

        courses_done = []
        stopping_words = ['become', 'becoming', 'career', 'interest',
'interested', 'cgpa', 'loved', 'goal']
        while (i < len(listt)):
            # stopping condition for courses done
            for stop in stopping_words:
                if listt[i] == stop:
                    return courses_done

            courses_done.append(listt[i])
            i += 1
        return courses_done
    except:
        raise Exception


def get_course_name(name):
    for course_name in list_of_courses:
        if name in course_name.lower():
            return course_name
    return 'None'


# ## Extracting Career Interest and Courses Done

# In[7]:


interest = ''
if 'engineer' in input_words:
    index = input_words.index('engineer')
    interest = find_career_interest(index, input_words)
print(interest)
```

```
courses_done = []
if 'courses' in input_words:
    index = input_words.index('courses')
    courses_done = find_courses_done(index, input_words)
# print(courses_done)


# ## Asserting Facts In a Text File

# In[8]:


facts_file = open("input_facts.txt", 'w')

if len(interest) > 0:
    assert_fact_interest = "interest('"+interest+"').\n"
    facts_file.write(assert_fact_interest)
else:
    assert_fact_interest = "interest('None').\n"
    facts_file.write(assert_fact_interest)


if len(courses_done) > 0:
    for name in courses_done:
        if get_course_name(name) == 'None':
            continue
        assert_course_done =
"course_taken('"+get_course_name(name)+"').\n"
        facts_file.write(assert_course_done)
else:
    assert_course_done = "course_taken('None').\n"
    facts_file.write(assert_course_done)


for i in range(len(interest_choices)):
    interest_choice_courses =
"interest_pre_requisite_courses('"+interest_choices[i]+"', ["
    if i == 0:
        for course in network_engineer_courses:
            interest_choice_courses += "'"+course+"', "
    if i == 1:
        for course in data_engineer_courses:
            interest_choice_courses += "'"+course+"', "
    if i == 2:
        for course in electronics_engineer_courses:
            interest_choice_courses += "'"+course+"', "
    if i == 3:
        for course in bioinformatics_engineer_courses:
            interest_choice_courses += "'"+course+"', "
    if i == 4:
        for course in robotics_engineer_courses:
            interest_choice_courses += "'"+course+"', "
    if i == 5:
        for course in ai_engineer_courses:
            interest_choice_courses += "'"+course+"', "
    if i == 6:
        for course in ml_engineer_courses:
            interest_choice_courses += "'"+course+"', "
    interest_choice_courses = interest_choice_courses[:-2]
    interest_choice_courses += "]).\n"

    facts_file.write(interest_choice_courses)


facts_file.close()


# In[9]:


# from pyswip import Prolog

# swipl = Prolog()
# swipl.consult("main.pl")
# # swipl.query("get_facts.")
# electives_suggested = list(swipl.query("start"))
# print(electives_suggested)
# # for i in range(len(electives_suggested)):
# #     var = electives_suggested[i].decode('utf-8')
# #     output(var)
```

[redacted]