CSE 643: Artificial Intelligence
Assignment 5

Meetakshi Setiya, 2019253

Implementation Details

I wrote a natural language interface in python using NLTK and Fuzzywuzzy. Below are the implementation details of the program.

- The natural language input is read from a file called input.txt and outputs (prolog facts) are dumped into a file called facts.pl.
- The program reads the input file line by line, collects them and sends them to a preprocessing module. Here, the string is converted to lowercase, tokenized using NLTK's word tokenizer and stop words and punctuations are removed. Parts of speech tagging is performed on these preprocessed tokens.
- Next, various facts are identified from these sets of tokens. Please note that since the nl interface is pretty rudimentary, certain constraints will have to be followed while writing the input plaintext sentences.
    - **Stream:** The program checks if out of the set of tokens made by a line there exists one which is a noun and is immediately followed by 'branch' or 'stream'. (E.g. I am in the CSE branch). The branch detected is then mapped to a code specified in the stream_code dictionary. If the user has not entered info about the stream in the input file, the program prompts for the stream code.
    - **Semester:** Semester is detected when there exists a token which is tagged CD (cardinal number) and is immediately preceded or followed by the word 'semester'. (E.g. I am currently in 7th semester). If the user has not entered info about the semester in the input file, the program prompts for it.
    - **CGPA:** CGPA is detected when there exists a token which is tagged CD (cardinal number) and is immediately preceded or followed by the word 'cgpa'. (E.g. My current CGPA is 9.6). If the user has not entered info about the CGPA in the input file, the program prompts for it.
    - **Credits (Core and Specialization):** Core credits are detected if a 'CD' (cardinal number) token is immediately followed by the word 'core-credits' and 'specialization-credits' or 'spzl-credits' for Specialization Credits. If the user has not entered info about the course credits, the program prompts for it. If the user is in the CS+X stream but has not entered info about the specialization credits, the program prompts for it. (Eg. I have done 20 core-credits and 16 specialization-credits.)

- **Interests:** If a token is tagged an adjective 'JJ' or the word is 'like' that means an interest is present. The name of the subject of interest is then retrieved by grouping all nouns at the end of a sentence together (along with some corner cases that had to be handled). (Eg. I really like AI/ML. I am interested in theoretical computer science). These interests are also converted to a code specified in the course_code dictionary. If no interests are detected, the program exists.
- **GPA and Courses Done:** If there is a token tagged 'CD' in the set preceded or followed by the word 'gpa', there exists a course done. To get this course the same reconstruction as interests is done but there were way too many corner cases to handle (stop words were also removed). So I used fuzzywuzzy to do a fuzzy search and get the best matching subject name using the reconstructed one. For best results, put a hyphen instead of space in the subject name. (E.g. I scored 10 GPA in Artificial Intelligence. I have an 8 GPA in Research-Methods-in-Social-Science-and-Design).
- These facts are collected and put into the facts.pl file. The head is suffixed with py for easy identification and no conflicts with the facts in the prolog program.

Integration with Assignment 1

Minor changes had to be made in the prolog file to integrate the nli. The facts.pl file is consulted by prolog. The user input fields had to go and instead of that, I used findall/3 to get the asserted facts. Handling Semester, CGPA, Core Credits, Interest List, Specialization Credits was trivial and the value obtained from findall/3 was simply passed to the recommend_all_courses/6 predicate.
For py_queried, py_type and py_grade I had to create an assert_facts/2 function to assert these facts to the working memory of the program. With these changes, the program was good to go.

Run the Code

1. Write the natural language input in input.txt file.
2. Run nli.py using the command *python nli.py.* The program might ask for inputs if some info for facts is wrong or missing in input.txt. The facts will be written to facts.pl
3. Run the main.pl file and get course recommendations.

Screenshots

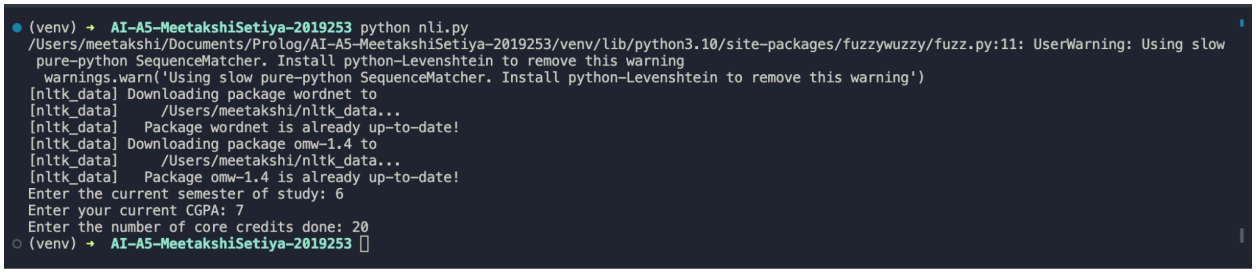
1.

Input:

input.txt

```
I am in CSE branch
my name is huahahahaha
I am interested in hardware and electronics
I am interested in Algorithms
I have 10 GPA in machine learning.
I have 8 GPA in computer-vision.
```

Output:



Note that the program asks for missing information

facts.pl

```
py_cgpa(7.0).
py_corecredits(20).
py_grade('Computer Vision', 8).
py_grade('Machine Learning', 10).
py_interests(3).
py_interests(8).
py_queried('Computer Vision').
py_queried('Machine Learning').
py_semester(6).
py_stream(1).
py_type('Computer Vision', 'Elective per Interest').
py_type('Machine Learning', 'Elective per Interest').
```

Prolog:

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

?- start.

-- ELECTIVES ADVISORY SYSTEM FOR 3RD & 4TH YEAR IIITD UNDERGRADUATES --

Hi! Please enter your information.
Name: met.


Great! met. Now, let me know your interests.


Let's narrow down your options.
Have you done Modern Algorithm Design ? (y/n) |: y.
What was your grade? |: 10.
Have you done Advanced Algorithms ? (y/n) |: n.
Have you done Introdiction to Graduate Algorithms ? (y/n) |: n.
Have you done Randomised Algorithms ? (y/n) |: n.
Have you done Concurrent and Learned Data Structures ? (y/n) |: y.
What was your grade? |: 8.
Have you done Approximation Algorithms ? (y/n) |: n.
Have you done Network Science ? (y/n) |: y.
What was your grade? |: 5.
Have you done Memory Testing and Design ? (y/n) |: n.
Have you done Digital Image Processing ? (y/n) |: y.
What was your grade? |: 10.
Have you done Wireless System Implementation ? (y/n) |: n.
Have you done Statistical Signal Processing ? (y/n) |: n.
Have you done Digital VLSI Design ? (y/n) |: n.
|: n.
Have you done Introduction to Nanoelectronics ? (y/n) |: y.
What was your grade? |: 6.
Have you done Control Theory ? (y/n) |: n.
Have you done Robotics ? (y/n) |: n.
Have you done Autonomous Driving ? (y/n) |: n.

Have you done Computer Networks ? (y/n) |: y.
What was your grade? |: 10.
Have you done Technical Communication ? (y/n) |: n.
Have you done Environmental Science ? (y/n) |: y.
What was your grade? |: 7.
```

```
Your course suggestions are ready, type get_suggestions.
true .

?-  get_suggestions.
Advanced Algorithms   |   Elective per Interest
true ;
Introdiction to Graduate Algorithms   |   Elective per Interest
true ;
Randomised Algorithms   |   Elective per Interest
true ;
Approximation Algorithms   |   Elective per Interest
true ;
Network Science   |   Elective per Interest, Available for Improvement
true ;
Memory Testing and Design   |   Elective per Interest
true ;
Wireless System Implementation   |   Elective per Interest
true ;
Statistical Signal Processing   |   Elective per Interest
true ;
Digital VLSI Design   |   Elective per Interest
true ;
Computer Architecture   |   Elective per Interest
true ;
Introduction to Nanoelectronics   |   Elective per Interest, Available for Improvement
true ;
Control Theory   |   Elective per Interest
true ;
Robotics   |   Elective per Interest
true ;
Autonomous Driving   |   Elective per Interest
true ;
Technical Communication   |   Department Mandatory
true.
```

2.

Input:

input.txt

```
Hi, I am Meetakshi
I am in CSAM branch
I have 8 GPA in Research Methods in Social Science and Design.
I have scored 9 gpa in Environmental-Science.
I am currently in 7th semester
I have a cgpa of 9.6
some random string that has no sense
I have done 34 core-credits
I have done 28 specialization-credits.
I am interested in AI/ML.
I also like theoretical computer science.
I have 10 gpa in foundation of computer security.
I have 10 GPA in machine learning.
another random string
hello world???
```

Output:

facts.pl

```
py_cgpa(9.6).
py_corecredits(34).
py_grade('Environmental Science', 9).
py_grade('Foundations Of Computer Security', 10).
py_grade('Machine Learning', 10).
py_grade('Research Methods in Social Science and Design', 8).
py_interests(1).
py_interests(4).
py_queried('Environmental Science').
py_queried('Foundations Of Computer Security').
py_queried('Machine Learning').
py_queried('Research Methods in Social Science and Design').
py_semester(7).
py_spzlcredits(28).
py_stream(3).
py_type('Environmental Science', 'Department Mandatory').
py_type('Foundations Of Computer Security', 'Elective per Interest').
py_type('Machine Learning', 'Elective per Interest').
py_type('Research Methods in Social Science and Design', 'Department Mandatory').
```

Prolog:

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

?- start.

-- ELECTIVES ADVISORY SYSTEM FOR 3RD & 4TH YEAR IIITD UNDERGRADUATES --

Hi! Please enter your information.
Name: met.

Great! met. Now, let me know your interests.


Let's narrow down your options.
Have you done Artificial Intelligence ? (y/n) |: n.
Have you done Natural Language Processing ? (y/n) |: n.
Have you done Reinforcement Learning ? (y/n) |: y.
What was your grade? |: 10.
Have you done Data Mining ? (y/n) |: n.
Have you done Meta Learning ? (y/n) |: n.
Have you done Edge AI ? (y/n) |: y.
What was your grade? |: 8.
Have you done Computer Vision ? (y/n) |: y.
What was your grade? |: 8.
Have you done Deep Learning ? (y/n) |: n.
Have you done Introduction to Quantum Computing ? (y/n) |: n.
Have you done Theory of Computation ? (y/n) |: n.
Have you done Program Verification ? (y/n) |: n.
Have you done Complexity Theory ? (y/n) |: n.
Have you done Program Analysis ? (y/n) |: n.
Have you done Decision Procedures ? (y/n) |: n.


Have you done BTech. Project ? (y/n) |: y.
What was your grade? |: 10.
Have you done Independent Project ? (y/n) |: y.
What was your grade? |: 5.
Have you done Independent Study ? (y/n) |: y.
What was your grade? |: 5.
Have you done Scientific Computing ? (y/n) |: y.
|: 8.
Have you done Probability and Random Processes ? (y/n) |: n.
Have you done Linear Optimization ? (y/n) |: n.
Have you done Statistical Inference ? (y/n) |: n.
Have you done Technical Communication ? (y/n) |: n.
```

```
Your course suggestions are ready, type get_suggestions.
true .

?- get_suggestions.
Artificial Intelligence   |   Elective per Interest
true ;
Natural Language Processing   |   Elective per Interest
true ;
Data Mining   |   Elective per Interest
true ;
Meta Learning   |   Elective per Interest
true ;
Deep Learning   |   Elective per Interest
true ;
Introduction to Quantum Computing   |   Elective per Interest
true ;
Theory of Computation   |   Elective per Interest
true ;
Program Verification   |   Elective per Interest
true ;
Complexity Theory   |   Elective per Interest
true ;
Program Analysis   |   Elective per Interest
true ;
Decision Procedures   |   Elective per Interest
true ;
Independent Project   |   Research, Available for Improvement
true ;
Independent Study   |   Research, Available for Improvement
true ;
Probability and Random Processes   |   Department Mandatory
true ;
Linear Optimization   |   Department Mandatory
true ;
Statistical Inference   |   Department Mandatory
true ;
Technical Communication   |   Department Mandatory
true.
```

Code

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from fuzzywuzzy import process
import string

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

facts = []
to_assert = {'stream': False, 'semester': False, 'cgpa': False, 'corecreds': False,
             'spzlcreds': False, 'interestlist': False, 'grade': False, 'queried':
False}

stream_code = {'cse': 1, 'ece': 2, 'csam': 3,
               'csd': 4, 'csai': 5, 'csb': 6, 'csss': 7}

course_code = {'ai ml': 1, 'ai/ml': 1, 'security cryptography': 2, 'algorithms': 3,
'theoretical computer science': 4, 'systems software development': 5,
               'mathematics': 6, 'computational biology': 7, 'hardware electronics':
8, 'design': 9, 'humanities social science': 10}

mandatory_courses = {1: ['Computer Networks', 'Technical Communication',
'Environmental Science'],
                     2: ['Digital Communication Systems', 'Digital Signal
Processig', 'Technical Communication', 'Environmental Science'],
                     3: ['Scientific Computing', 'Probability and Random Processes',
'Linear Optimization', 'Statistical Inference', 'Technical Communication',
'Environmental Science'],
                     4: ['Computer Networks', 'Research Methods in Social Science
and Design',
                         'Design of Interactive Systems', 'Technical Communication',
'Environmental Science'],
                     5: ['Machine Learning', 'Computer Networks', 'Artificial
Intelligence', 'Ethics in AI',
                         'Technical Communication', 'Environmental Science'],
                     6: ['Biophysics', 'Algorithms in Computational Biology',
'Technical Communication', 'Environmental Science'],
                     7:  ['Computer Networks', 'Technical Communication',
'Environmental Science']}

elective_courses = {
   1: ['Artificial Intelligence', 'Machine Learning', 'Natural Language
Processing',
```

```
       'Reinforcement Learning', 'Data Mining', 'Meta Learning', 'Edge AI',
       'Computer Vision', 'Deep Learning'],
   2: ['Foundations Of Computer Security', 'Network Anonymity & Privacy', 'Applied
Cryptography',
       'Topics In Adaptive Cybersecurity', 'Topics In Cryptanalysis', 'Networks And
System Security II',
       'Advanced Biometrics'],
   3: ['Modern Algorithm Design', 'Advanced Algorithms', 'Introduction to Graduate
Algorithms',
       'Randomised Algorithms', 'Concurrent and Learned Data Structures',
'Approximation Algorithms',
       'Network Science'],
   4: ['Introduction to Quantum Computing', 'Theory of Computation', 'Program
Verification',
       'Complexity Theory', 'Program Analysis', 'Decision Procedures'],
   5: ['Distributed Systems: Concepts & Design', 'Parallel Runtimes For Modern
Processors',
       'Compilers', 'Advanced Operating Systems', 'Programmable Networking',
'Mobile Computing',
       'Software Development Using Open-Source', 'Multimedia Computing &
Applications',
       'Systems Analysis, Design & Requirements Engineering'],
   6: ['Linear Optimization', 'Advanced Linear Algebra', 'Calculus in R^N',
'Scientific Computing',
       'Complex Analysis', 'Algebraic Number Theory', 'Finite & Spectral Element
Methods',
       'Categorical Data Analysis'],
   7: ['Computational Gastronomy', 'Algorithms in Bioinformatics', 'Foundations of
Modern Biology',
       'Biomedical Image Processing', 'Introduction to Mathematical Biology',
'Biophysics',
       'Data Science for Genomics', 'Systems and Synthetic Biology'],
   8: ['Memory Testing and Design', 'Digital Image Processing', 'Wireless System
Implementation',
       'Statistical Signal Processing', 'Digital VLSI Design', 'Computer
Architecture',
       'Introduction to Nanoelectronics', 'Control Theory', 'Robotics', 'Autonomous
Driving'],
   9: ['Design of Interactive Systems', 'Human Centred AI', 'Introduction To Motion
Graphics',
       'Introduction To Animation And Graphics', 'Game Design & Development',
'Introduction To 3D Animation',
       'Fundamentals Of Audio For Engineers', 'Advanced Topics In Human Centered
Computing'],
   10: ['Game Theory', 'Foundations Of Finance', 'Industrial Organization', 'Ethics
in AI',
        'Learning and Memory', 'Social Psychology', 'Entrepreneurial Kichadi',
```

```
'Philosophy of Mind',
        'Intersectionality Studies', 'Advanced Ethnographic Research Methods'],
    11: ['BTech. Project', 'Independent Project', 'Independent Study']
}


def get_semester(line):
    global facts
    if to_assert['semester']:
        return
    for i in range(len(line)-1):
        word1, pos1 = line[i]
        word2, pos2 = line[i+1]
        if (pos1 == 'CD' and word2 == 'semester'):
            word1 = word1[:1]  # just the number
            fact = f"py_semester({word1})"
            facts.append(fact)
            to_assert['semester'] = True
        elif (pos2 == 'CD' and word1 == 'semester'):
            word2 = word2[:1]
            fact = f"py_semester({word2})"
            facts.append(fact)
            to_assert['semester'] = True


def get_stream(line):
    global facts, stream_code
    if to_assert['stream']:
        return
    for i in range(len(line)-1):
        word1, pos1 = line[i]
        word2, _ = line[i+1]
        if (pos1 == 'NN') and (word2 == 'branch' or word2 == 'stream'):
            code = stream_code[word1]
            fact = f"py_stream({code})"
            facts.append(fact)
            to_assert['stream'] = code


def get_cgpa(line):
    global facts
    if to_assert['cgpa']:
        return
    for i in range(len(line)-1):
        word1, pos1 = line[i]
        word2, pos2 = line[i+1]
        if pos1 == 'CD' and word2 == 'cgpa':
```

```
            fact = f"py_cgpa({word1})"
            facts.append(fact)
            to_assert['cgpa'] = True
        if pos2 == 'CD' and word1 == 'cgpa':
            fact = f"py_cgpa({word2})"
            facts.append(fact)
            to_assert['cgpa'] = True


def get_credits(line):
    global facts
    if to_assert['corecreds'] and to_assert['spzlcreds']:
        return
    for i in range(len(line)-1):
        word1, pos1 = line[i]
        word2, _ = line[i+1]
        if pos1 == 'CD' and word2 == 'core-credits':
            fact = f"py_corecredits({word1})"
            facts.append(fact)
            to_assert['corecreds'] = True
        if pos1 == 'CD' and (word2 == 'specialization-credits' or word2 ==
'spzl-credits'):
            fact = f"py_spzlcredits({word1})"
            facts.append(fact)
            to_assert['spzlcreds'] = True


def get_interests(line):
    global facts
    if "gpa" in dict(line):
        return
    for i in line:
        word, pos = i
        if pos == "JJ" or word == "like":
            course = get_interest_name(line)
            to_assert['interestlist'] = True
            if course:
                code = course_code[course]
                fact = f"py_interests({code})"
                facts.append(fact)
                return


def get_gpa_courses_done(line):
    global facts
    for i in range(len(line)-1):
        word1, pos1 = line[i]
```

```
        word2, pos2 = line[i+1]
        if (pos1 == 'CD' and word2 == 'gpa'):
            intr = get_interest_name(line)
            if (intr):
                c, t = get_course_name(intr)
                facts.append(f"py_grade('{c}', {word1})")
                facts.append(f"py_queried('{c}')")
                facts.append(f"py_type('{c}', '{t}')")
        elif (pos2 == 'CD' and word1 == 'gpa'):
            intr = get_interest_name(line)
            if (intr):
                c, t = get_course_name(intr)
                facts.append(f"py_grade('{c}', '{word2}')")
                facts.append(f"py_queried('{c}')")
                facts.append(f"py_type('{c}', '{t}')")


def get_course_name(course):
    global facts
    maxscore = 0
    bestcourse = ""
    besttype = ""
    if to_assert['stream']:
        for k in elective_courses.keys():
            courselist = elective_courses[k]
            c, s = process.extractOne(course, courselist)
            if s > maxscore:
                maxscore = s
                bestcourse = c
                besttype = "Elective per Interest"
        for k in mandatory_courses.keys():
            courselist = mandatory_courses[k]
            c, s = process.extractOne(course, courselist)
            if s >= maxscore:
                maxscore = s
                bestcourse = c
                if not besttype == "Department Mandatory":
                    besttype = "Elective per Interest"
                if k == to_assert['stream']:
                    besttype = "Department Mandatory"
        return bestcourse, besttype


def get_interest_name(line):
    intr = ""
    for i in range(len(line)-1, -1, -1):
        (word, pos) = line[i]
```

```
        if pos == "NN" or pos == "NNP" or pos == "NNS":
            intr = f"{word} {intr}"
        elif word == "theoretical" and intr.strip() == "computer science":
            intr = f"{word} {intr}"
        elif word == "computational" and intr.strip() == "biology":
            intr = f"{word} {intr}"
        elif word == "social" and "science" in intr.strip():
            intr = f"{word} {intr}"
        else:
            break
    if len(intr) > 0:
        return intr.strip()


def pos_tagging(tokenised_lines):
    tagged = []
    for l in tokenised_lines:
        tagged.append(nltk.pos_tag(l))
    return tagged


def preprocess(lines):
    # tokenize and remove stop words
    stop_words = set(stopwords.words('english'))
    tokenized_lines = []
    for line in lines:
        line = line.lower().strip()
        tokens = word_tokenize(line)
        cleaned_tokens = [t for t in tokens if not t in stop_words]
        cleaned_tokens = list(
            filter(lambda t: t not in string.punctuation, cleaned_tokens))
        tokenized_lines.append(cleaned_tokens)
    return tokenized_lines


def make_facts(tagged_lines):
    for i in tagged_lines:
        get_stream(i)

    if not to_assert['stream']:
        st = int(input("Enter your stream code: "))
        fact = f"py_stream({st})"
        facts.append(fact)
        to_assert['stream'] = st

    for i in tagged_lines:
        get_semester(i)
```

```
        get_cgpa(i)
        get_credits(i)
        get_interests(i)
        get_gpa_courses_done(i)
    # semester, gpa etc also'
    if not to_assert['semester']:
        sem = int(input("Enter the current semester of study: "))
        fact = f"py_semester({sem})"
        facts.append(fact)
        to_assert['semester'] = True

    if not to_assert['cgpa']:
        cg = float(input("Enter your current CGPA: "))
        fact = f"py_cgpa({cg})"
        facts.append(fact)
        to_assert['cgpa'] = True

    if to_assert['stream'] and to_assert['stream'] > 2:
        if not to_assert['spzlcreds']:
            spc = int(input("Enter the number of specialization credits done: "))
            fact = f"py_spzlcredits({spc})"
            facts.append(fact)
            to_assert['spzlcreds'] = True

    if not to_assert['corecreds']:
        cc = int(input("Enter the number of core credits done: "))
        fact = f"py_corecredits({cc})"
        facts.append(fact)
        to_assert['corecreds'] = True

    assert to_assert['interestlist'] == True, "Enter some interests and restart."


def main():
    lines = []
    with open("input.txt") as fileptr:
        lines = fileptr.readlines()
    preprocessed_lines = preprocess(lines)
    tagged_lines = pos_tagging(preprocessed_lines)
    make_facts(tagged_lines)
    with open("facts.pl", "w") as fileptr:
        facts.sort()
        for i in facts:
            fileptr.write(f"{i}.\n")


main()
```

[redacted]