CSE643: Artificial Intelligence
Assignment-3

- Created a python program using a durable-rules module with forward-chaining rules for course-and -extracurricular activities suggestion system

- The program suggests possible options for a student based on forward-chaining rules.

- Code:

```
# forward chaining rule engine
# trying out durable rules engine
from durable.lang import *

with ruleset('interests'):
   # will be triggered by 'interests' facts

   ##Different hobies based on interest in mathematics and Linear Algebra
   @when_all((m.area == 'mathematical') & (m.type == 'Linear-Algebra') & (m.hobby ==
'programming'))
   def mp(c):
       c.assert_fact('skills', { 'field': 'Linear-Algebra' })
       c.assert_fact('preferences', { 'type': 'mltheory' })
       c.assert_fact('hobby', { 'category': 'programming' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })

   @when_all((m.area == 'mathematical') & (m.type == 'Linear-Algebra') & (m.hobby ==
'designing'))
   def mp(c):
       c.assert_fact('skills', { 'field': 'Linear-Algebra' })
       c.assert_fact('preferences', { 'type': 'mltheory' })
       c.assert_fact('hobby', { 'category': 'designing' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })

   @when_all((m.area == 'mathematical') & (m.type == 'Linear-Algebra') & (m.hobby ==
'fashion'))
   def mp(c):
       c.assert_fact('skills', { 'field': 'Linear-Algebra' })
       c.assert_fact('preferences', { 'type': 'mltheory' })
       c.assert_fact('hobby', { 'category': 'fashion' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
```

```
'robotics' })

   @when_all((m.area == 'mathematical') & (m.type == 'Linear-Algebra') & (m.hobby ==
'comedy'))
   def mp(c):
       c.assert_fact('skills', { 'field': 'Linear-Algebra' })
       c.assert_fact('preferences', { 'type': 'mltheory' })
       c.assert_fact('hobby', { 'category': 'comedy' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })

   @when_all((m.area == 'mathematical') & (m.type == 'Linear-Algebra') & (m.hobby ==
'music'))
   def mp(c):
       c.assert_fact('skills', { 'field': 'Linear-Algebra' })
       c.assert_fact('preferences', { 'type': 'mltheory' })
       c.assert_fact('hobby', { 'category': 'music' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })


   ##Different hobies based on interest in non-mathematics and programming
   @when_all((m.area == 'non-mathematical') & (m.type == 'programming') & (m.hobby ==
'music'))
   def nmp(c):
       c.assert_fact('skills', { 'field': 'programming' })
       c.assert_fact('preferences', { 'type': 'non-theory' })
       c.assert_fact('hobby', { 'category': 'music' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advanced Programming' })

   @when_all((m.area == 'non-mathematical') & (m.type == 'programming') & (m.hobby ==
'programming'))
   def nmp(c):
       c.assert_fact('skills', { 'field': 'programming' })
       c.assert_fact('preferences', { 'type': 'non-theory' })
       c.assert_fact('hobby', { 'category': 'programming' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advanced Programming' })

   @when_all((m.area == 'non-mathematical') & (m.type == 'programming') & (m.hobby ==
'comedy'))
```

```
   def nmp(c):
       c.assert_fact('skills', { 'field': 'programming' })
       c.assert_fact('preferences', { 'type': 'non-theory' })
       c.assert_fact('hobby', { 'category': 'comedy' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advanced Programming' })

   @when_all((m.area == 'non-mathematical') & (m.type == 'programming') & (m.hobby ==
'designing'))
   def nmp(c):
       c.assert_fact('skills', { 'field': 'programming' })
       c.assert_fact('preferences', { 'type': 'non-theory' })
       c.assert_fact('hobby', { 'category': 'designing' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advanced Programming' })

   @when_all((m.area == 'non-mathematical') & (m.type == 'programming') & (m.hobby ==
'fashion'))
   def nmp(c):
       c.assert_fact('skills', { 'field': 'programming' })
       c.assert_fact('preferences', { 'type': 'non-theory' })
       c.assert_fact('hobby', { 'category': 'fashion' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advanced Programming' })


   ##Different hobies based on interest in theory and non-programming
   @when_all((m.area == 'theory') & (m.type == 'non-programming') & (m.hobby ==
'designing'))
   def nmp(c):
       c.assert_fact('skills', { 'field': 'non-programming' })
       c.assert_fact('hobby', { 'category': 'designing' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advance OS' })

   @when_all((m.area == 'theory') & (m.type == 'non-programming') & (m.hobby ==
'music'))
   def nmp(c):
       c.assert_fact('skills', { 'field': 'non-programming' })
       c.assert_fact('hobby', { 'category': 'music' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advance OS' })
```

```
   @when_all((m.area == 'theory') & (m.type == 'non-programming') & (m.hobby ==
'comedy'))
   def nmp(c):
        c.assert_fact('skills', { 'field': 'non-programming' })
        c.assert_fact('hobby', { 'category': 'comedy' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advance OS' })

   @when_all((m.area == 'theory') & (m.type == 'non-programming') & (m.hobby ==
'fashion'))
   def nmp(c):
        c.assert_fact('skills', { 'field': 'non-programming' })
        c.assert_fact('hobby', { 'category': 'fashion' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advance OS' })

   @when_all((m.area == 'theory') & (m.type == 'non-programming') & (m.hobby ==
'programming'))
   def nmp(c):
        c.assert_fact('skills', { 'field': 'non-programming' })
        c.assert_fact('hobby', { 'category': 'programming' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advance OS' })


   ##Different hobies based on interest in human-like-conversations and notheory
   @when_all((m.area == 'human-like-conversations') & (m.type == 'notheory') &
(m.hobby == 'programming'))
   def hlcnt(c):
        c.assert_fact('skills', { 'field': 'probability' })
        c.assert_fact('hobby', { 'category': 'programming' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'natural language processing' })

   @when_all((m.area == 'human-like-conversations') & (m.type == 'notheory') &
(m.hobby == 'designing'))
   def hlcnt(c):
        c.assert_fact('skills', { 'field': 'probability' })
        c.assert_fact('hobby', { 'category': 'designing' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'natural language processing' })
```

```
   @when_all((m.area == 'human-like-conversations') & (m.type == 'notheory') &
(m.hobby == 'music'))
   def hlcnt(c):
        c.assert_fact('skills', { 'field': 'probability' })
        c.assert_fact('hobby', { 'category': 'music' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'natural language processing' })

   @when_all((m.area == 'human-like-conversations') & (m.type == 'notheory') &
(m.hobby == 'comedy'))
   def hlcnt(c):
        c.assert_fact('skills', { 'field': 'probability' })
        c.assert_fact('hobby', { 'category': 'comedy' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'natural language processing' })

   @when_all((m.area == 'human-like-conversations') & (m.type == 'notheory') &
(m.hobby == 'fashion'))
   def hlcnt(c):
        c.assert_fact('skills', { 'field': 'probability' })
        c.assert_fact('hobby', { 'category': 'fashion' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'natural language processing' })


   ##Different hobies based on interest in human-like-conversations and theory
   @when_all((m.area == 'human-like-conversations') & (m.type == 'theory') & (m.hobby
== 'comedy'))
   def hlct(c):
        c.assert_fact('skills', { 'field': 'probability' })
        c.assert_fact('hobby', { 'category': 'comedy' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advanced Programming' })
        c.assert_fact('preferences', { 'type': 'non-theory' })

   @when_all((m.area == 'human-like-conversations') & (m.type == 'theory') & (m.hobby
== 'fashion'))
   def hlct(c):
        c.assert_fact('skills', { 'field': 'probability' })
        c.assert_fact('hobby', { 'category': 'fashion' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
```

```
'Advanced Programming' })
        c.assert_fact('preferences', { 'type': 'non-theory' })

    @when_all((m.area == 'human-like-conversations') & (m.type == 'theory') & (m.hobby
== 'designing'))
    def hlct(c):
        c.assert_fact('skills', { 'field': 'probability' })
        c.assert_fact('hobby', { 'category': 'designing' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advanced Programming' })
        c.assert_fact('preferences', { 'type': 'non-theory' })

    @when_all((m.area == 'human-like-conversations') & (m.type == 'theory') & (m.hobby
== 'programming'))
    def hlct(c):
        c.assert_fact('skills', { 'field': 'probability' })
        c.assert_fact('hobby', { 'category': 'programming' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advanced Programming' })
        c.assert_fact('preferences', { 'type': 'non-theory' })

    @when_all((m.area == 'human-like-conversations') & (m.type == 'theory') & (m.hobby
== 'music'))
    def hlct(c):
        c.assert_fact('skills', { 'field': 'probability' })
        c.assert_fact('hobby', { 'category': 'music' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'Advanced Programming' })
        c.assert_fact('preferences', { 'type': 'non-theory' })


    ##Different hobies based on interest in human-like-interactions and theory
    @when_all((m.area == 'human-like-interactions') & (m.type == 'theory') & (m.hobby
== 'programming'))
    def hlit(c):
        c.assert_fact('skills', { 'field': 'ai-ml' })
        c.assert_fact('hobby', { 'category': 'programming' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })
        c.assert_fact('preferences', { 'type': 'mltheory' })

    @when_all((m.area == 'human-like-interactions') & (m.type == 'theory') & (m.hobby
```

```
== 'designing'))
   def hlit(c):
       c.assert_fact('skills', { 'field': 'ai-ml' })
       c.assert_fact('hobby', { 'category': 'designing' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })
       c.assert_fact('preferences', { 'type': 'mltheory' })

   @when_all((m.area == 'human-like-interactions') & (m.type == 'theory') & (m.hobby
== 'music'))
   def hlit(c):
       c.assert_fact('skills', { 'field': 'ai-ml' })
       c.assert_fact('hobby', { 'category': 'music' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })
       c.assert_fact('preferences', { 'type': 'mltheory' })

   @when_all((m.area == 'human-like-interactions') & (m.type == 'theory') & (m.hobby
== 'comedy'))
   def hlit(c):
       c.assert_fact('skills', { 'field': 'ai-ml' })
       c.assert_fact('hobby', { 'category': 'comedy' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })
       c.assert_fact('preferences', { 'type': 'mltheory' })

   @when_all((m.area == 'human-like-interactions') & (m.type == 'theory') & (m.hobby
== 'fashion'))
   def hlit(c):
       c.assert_fact('skills', { 'field': 'ai-ml' })
       c.assert_fact('hobby', { 'category': 'fashion' })
       c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })
       c.assert_fact('preferences', { 'type': 'mltheory' })


   ##Different hobies based on interest in human-like-interactions and theory
   @when_all((m.area == 'human-like-interactions') & (m.type == 'notheory') & (m.hobby
== 'fashion'))
   def hlint(c):
       c.assert_fact('skills', { 'field': 'ai-ml' })
       c.assert_fact('hobby', { 'category': 'fashion' })
```

```
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })

    @when_all((m.area == 'human-like-interactions') & (m.type == 'notheory') & (m.hobby
== 'comedy'))
    def hlint(c):
        c.assert_fact('skills', { 'field': 'ai-ml' })
        c.assert_fact('hobby', { 'category': 'comedy' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })

    @when_all((m.area == 'human-like-interactions') & (m.type == 'notheory') & (m.hobby
== 'programming'))
    def hlint(c):
        c.assert_fact('skills', { 'field': 'ai-ml' })
        c.assert_fact('hobby', { 'category': 'programming' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })

    @when_all((m.area == 'human-like-interactions') & (m.type == 'notheory') & (m.hobby
== 'designing'))
    def hlint(c):
        c.assert_fact('skills', { 'field': 'ai-ml' })
        c.assert_fact('hobby', { 'category': 'designing' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })

    @when_all((m.area == 'human-like-interactions') & (m.type == 'notheory') & (m.hobby
== 'music'))
    def hlint(c):
        c.assert_fact('skills', { 'field': 'ai-ml' })
        c.assert_fact('hobby', { 'category': 'music' })
        c.assert_fact({ 'subject': 'choose', 'predicate': 'elective', 'object':
'robotics' })

    @when_all(+m.subject)
    def output(c):
        print('Fact: {0} {1} {2}'.format(c.m.subject, c.m.predicate, c.m.object))

## Rule-Suggestions based on skills
with ruleset('skills'):
    @when_all((m.field == 'Linear-Algebra'))
```

```
    def la(d):
        d.assert_fact({ 'subject': 'take ML course' })
        d.assert_fact({ 'subject': 'take DIP course' })
        d.assert_fact({ 'subject': 'take Data Mining course' })

    @when_all((m.field == 'programming'))
    def prog(d):
        d.assert_fact({ 'subject': 'take FPP course' })
        d.assert_fact({ 'subject': 'take Compilers course' })
        d.assert_fact({ 'subject': 'take Data Science course' })

    @when_all((m.field == 'non-programming'))
    def prog(d):
        d.assert_fact({ 'subject': 'take OS course' })
        d.assert_fact({ 'subject': 'take DBMS course' })
        d.assert_fact({ 'subject': 'take CN course' })

    @when_all((m.field == 'probability'))
    def prob(d):
        d.assert_fact({ 'subject': 'take Probability and Statistics course' })

    @when_all((m.field == 'ai-ml'))
    def aiml(d):
        d.assert_fact({ 'subject': 'take AI course' })
        d.assert_fact({ 'subject': 'take ML course' })
        d.assert_fact( {'subject': 'take DL course' })

    @when_all(+m.subject)
    def output(d):
        print('Fact: {0}'.format(d.m.subject))

## Suggestions based on preferences
with ruleset('preferences'):
    @when_all((m.type == 'mltheory'))
    def mathct(e):
        e.assert_fact({ 'subject': 'take DL theory course'})

    @when_all((m.type == 'theory'))
    def mathct(e):
        e.assert_fact({ 'subject': 'take DSA course'})

    @when_all((m.type == 'non-theory'))
```

```
    def mathct(e):
        e.assert_fact({ 'subject': 'take OOPD course'})

    @when_all((m.type == 'nltheory'))
    def mathct(e):
        e.assert_fact({ 'subject': 'take NLP foundations course'})

    @when_all(+m.subject)
    def output(c):
        print('Fact: {0}'.format(c.m.subject))

## club details based on hobbies
with ruleset('hobby'):
    @when_all((m.category == 'fashion'))
    def funa(f):
        f.assert_fact({ 'subject': 'join club MUSE'})

    @when_all((m.category == 'designing'))
    def funa(f):
        f.assert_fact({ 'subject': 'join club BIOBYTES'})

    @when_all((m.category == 'programming'))
    def funa(f):
        f.assert_fact({ 'subject': 'join club FOOBAR PROGRAMMING CLUB'})

    @when_all((m.category == 'music'))
    def funa(f):
        f.assert_fact({ 'subject': 'join club AUDIOBYTES'})

    @when_all((m.category == 'comedy'))
    def funa(f):
        f.assert_fact({ 'subject': 'join club STAND-UP Comedy'})

    @when_all(+m.subject)
    def output(f):
        print('Fact: {0}'.format(f.m.subject))
```

- OUTPUTS

```
##Uncomment a sentence to execute
#assert_fact('interests', { 'area': 'mathematical', 'type': 'Linear-Algebra', 'hobby':
'music' })
```

```
================================ RESTART: /Users/yuktigoswami/Desktop/ai_assignment_3.py ==
Fact: take Data Mining course
Fact: take DIP course
Fact: take ML course
Fact: take DL theory course
Fact: join club AUDIOBYTES
Fact: choose elective robotics
>>>
```

```
## changing hobbies for same interests
#assert_fact('interests', { 'area': 'theory', 'type': 'non-programming', 'hobby':
'designing' })
```

```
============ RESTART: /Users/yuktigoswami/Desktop/ai_assignment_3.py ===========
Fact: take CN course
Fact: take DBMS course
Fact: take OS course
Fact: join club BIOBYTES
Fact: choose elective Advance OS
```

```
#assert_fact('interests', { 'area': 'theory', 'type': 'non-programming', 'hobby':
'music' })
```

```
============ RESTART: /Users/yuktigoswami/Desktop/ai_assignment_3.py ===========
Fact: take CN course
Fact: take DBMS course
Fact: take OS course
Fact: join club AUDIOBYTES
Fact: choose elective Advance OS
```