Artificial Intelligence (Spring 2014)
Project-II
Phase 1

(Draft 2)

Introduction:

In this exercise, you will be using the NLP toolkit : "**nltk**" in python, to implement a First order Predicate Logic based "query prover" and "question answering system" for a recent cricket ODI series dataset.

Before starting, you need to install "nltk".
If you are using linux, you can use the following to install it:

sudo apt-get install python-nltk

Read the "Explanation" section completely and understand the associated codes. The tasks that you have to perform are given in the "Tasks to perform" section.

The deadline for all the tasks is **01-03-2014, 5:00 PM.** It is highly recommended to code in python, otherwise you will have to figure out NLP toolkits for other language like Java. In case of any doubts, post them in "courses portal".

Files included:

The uploaded folder in the "Resources" section of courses portal contains the following:

```
project2/
   |-------- Dataset/
   |              |--------- match2/ : contains files for 2nd ODI match
   |              |--------   player_profile/ : contains files for player profiles (columns are tab
separated)
   |-------- sample_solver.py : This file contains the sample code for the "solver" for a limited
dataset and a given query.
   |
   |-------- basic_code1.py : supplement code for "STEP 1" of Explanation section
   |
   |-------- basic_code2.py : supplement code for "STEP 3" of Explanation section
```

Explanation:

Let us proceed from the very basics.

STEP - 1 : Understanding how to use "nltk" for first order predicate logic.

We need to first look at how we can use some of the functionalities provided by "nltk" to:

a. Represent given set of predicates
b. Represent the given query to be proved/disproved
c. Use the corresponding functions for proving.

As an example, let us consider the following scenario (open the code "*basic_code1.py"*):
Given data:

- Sachin scored a ton.
- If a player score a ton or takes a five-wicket haul, then he is given "man of match" award.

Query:

- Sachin gets the "man of match" award.

So for this problem, we first identify the predicates as :

ton(x) : x scores a ton.
momatch(x) : x gets man of match award,
fwhaul(x): x takes a five wicket haul

Now the given predicates become:

- ton(sachin)
- for all x, ((ton(x) or fwhaul(x)) implies momatch(x))

And the query becomes :

- momatch(sachin)

Now look at the code "basic_code1.py" to understand how we represent predicates in proper form.

For further understanding of the functions described in the code and getting familiarity with the syntax look at:

- http://www.nltk.org/howto/inference.html
- http://www.nltk.org/book/ch10.html

STEP 2: Understanding the dataset

The dataset to be used for this project, will consist of data from a 5-match cricket ODI series (taken from www.espncricinfo.com). Since the amount of data which is considered is not huge, you can use in memory data structures which will easily fit in main memory. (As discussed in the last tutorial, one could also use "MySQLdb" module in python to use MySQL for storing and retrieving data, but learning to use this module would take more time).

The links to the 5 matches are:

1. http://www.espncricinfo.com/new-zealand-v-india-2014/engine/match/667641.html
2. http://www.espncricinfo.com/new-zealand-v-india-2014/engine/match/667643.html
3. http://www.espncricinfo.com/new-zealand-v-india-2014/engine/match/667645.html
4. http://www.espncricinfo.com/new-zealand-v-india-2014/engine/match/667647.html
5. http://www.espncricinfo.com/new-zealand-v-india-2014/engine/match/667649.html

For this phase, data to be considered is as follows:

A. For each of the 5 matches
    1. "Scorecard" information (both batting and bowling figures for both sides)
    2. The player of the match information
    3. The winning team name
B. For the entire series:
    1. Player information for both sides.

We have provided the "Scorecard" information ONLY FOR the second ODI. These files are "," separated and can be parsed using the split() function.

But for the other ODIs, you need to first BUILD the required files (look at the files in the folder *./dataset/match2/*) in a format which you can easily parse later.

The player information for both teams IS PROVIDED (so you do not have to make files for that). Look at the 2 files in "*./dataset/player_profile/*" *,* these have "TAB" separated columns and the information is in the following form:

<player ID>, <player full name>, <Date of birth, place of birth>, <age>, <teams for which player plays>, <playing role>, <batting style>, <bowling style>

<u>**STEP 3**: Understanding what has to be done by looking at an example:</u>

Before proceeding to the following section, look at the code *basic_code2.py.* You can understand the code based on the comments provided. Make sure you are clear and understand the code before proceeding.

Let us consider an example, where the dataset consists of only the “Scorecard” information of the second ODI match. And, we are given the following query to prove or disprove (for this match):

All players who have strike rate above 150.0 have hit at least 1 six (more than 0 six(es)) in the match.

We identify the predicates as:

- srate(x) : player x has strike rate greater than 150.0
- gtsix(x) : player x has hit greater than 0 six(es).

Look at the code - “**sample_solver.py**” for the implementation details.

Few points regarding implementation:

- The data is read from the 4 files and stored in 2 dictionaries - bats, bowl (but since this is only for 1 match, you need to think about better ways of handling data in case of the complete dataset).
- For solving the query, 2 functions “parse_for_six”, “parse_for_sr” had to be created to retrieve the required player data.
- Since, proper formatted string is required to create a model (based on assumptions), temporary strings are created.
- Lastly, the model is built and query is evaluated. Also results have been shown which satisfy the condition of the query.

So, you will be given 20 queries, for which you will have to write approximately 40-50 functions (for generating predicates), and construct appropriate model (based on the dataset) and prove/disprove each of the queries. Also, you need to print answers for queries, which are true or which explicitly ask you to print the results (satisfying that particular query).

The 20 queries are divided into 2 sets (A and B). You have to write functions for answering all the queries.

SET-A: Prove or disprove. If the output is yes, print the values which satisfy the predicate

1. For all matches, “Player of match” award is given to player of winning team.
2. For all matches, losing side consists of at least 1 ducks in the batting innings.
3. For all innings, if strike rate of player is above 200.0 then he has hit more sixes than fours.
4. For all matches, winning side contains at least 1 player who hit at least 1 boundary(no. of 4s) yet his strike rate was below 100.
5. There exists player(s) in the series, who have scored more than 50 runs in batting and claimed at least 1 wicket (1 or more) in bowling, in the same match.
6. For all matches, for any side, there exist at least 1 bowler who has bowled more than 7

overs and failed to get any wicket.

7. In any of the matches, there exists a bowler who did not claim any wicket and went for more than 8 runs per over (economy > 8.00).
8. There exists a match, where a batsman scored hundred and despite that the team lost.
9. For all matches, right handed bowlers claim more wickets than left handed bowlers. (you will have to use info. from player_profile data)
10. There exists a player, who is less than 26 years old and has scored more than 250 runs in the series without any ducks in any of the matches. (you will have to use info from player_profile data)

SET-B: Question answer based and few are open-ended and require building of heuristics

1. Who are the players who played all matches in the series ?
2. Did Ishant sharma bowl more wides than Sir Jadeja ?
3. Did Southee take more "catches" than Ryder ?
4. Is there a player who has been awarded player of the match twice ?
5. Did Sir Jadeja bowl better in innings1 or innings 2 ?
6. Prove that Dhoni is a hard-hitting batsman. (You can create your own heuristic that tries to identify what hard hitting could mean; It can mean high strike rate, more sixes, less dots and so on, more interesting heuristics get bonus marks !)
7. Is Ishant sharma a better bowler than Sir Jadeja ? (You can again create a function that tries to capture what better may mean in this case, for some of you it might mean - more wickets, for some it may mean a better economy rate, interesting heuristics get bonus marks !)
8. Do the middle order batsmen perform better than the opening batsmen in general ? (Design your heuristic carefully, considering all the middle order batsmen vs all the opening batsmen)
9. Do the teams that win matches win tosses too ?
10. Based on given matches, what do you predict as the outcome of next match ? (Take into consideration the fact that players are also changing between the matches, and hence the teams are changing.)

Tasks to perform:

1. Building the rest of the database as described in STEP-2 of "Explanation" section.
2. Writing code for answering the 20 queries given in STEP-3 of "Explanation" section. (you can use the *sample_solver.py* as the starting point and build your code over it).

Grading and Evaluation:

1. For part(1), 5 marks.
2. For part(2), 20 + 20 = 40 marks. (for each of the 20 queries, the correct answer, functions written, reusability of code, and heuristics used (for specific queries) will be considered).