Problem: Design a Rule Based NLU – Pattern Matching Algorithm

**Conditions**:

Deliverables from Participants:

* Submission should be in a zip file [zip filename should be with the number of group e.g. **group1.zip**].
* The zip should contain following items.

(1) “Testing/output.txt” [there is a folder name “Testing” participants need to create output.txt in that folder]

(2) Source Code [samsungtest.c or samsungtest.cpp or samsungtest.java]

(3) All dependent files (we are providing resource files with Problem statement, you need to keep those resource files as is in zip. If you create your own extra resource file – include that also in zip.)

(4) A "readme.txt", which will include the brief write-up about your approach.

(5) Executable (exe or a.out] of the program.

Details for Submission:

- Submitted code should compile and execute from any directory during evaluation. No absolute path should be used in your code.

- No external library should be used. Only use inbuilt library.

- Program should run in below steps during evaluation

- Source code should be compliable in Linux system (Windows user can use cygwin).

**Java (less than or equal to 1.7) programming language**

Step 1 : Go to folder "groupx"

Step 2 : > javac samsungtest.java

Step 3 : > java samsungtest

**C programming language**

Step 1 : Go to folder "groupx"

Step 2 : gcc samsungtest.c [or samsungtest.cpp]

Step 3 : ./a.out

**Note**:

- Filename for output and source code and all other files must be as per instruction.

If any one of them is different, then the submission will be rejected.

**Changes from Previous Question:**

**Problem description changes**: - All the problem description changes are marked in **Green** color font.

**Resource files changes**:-

1. Added three more Grammar files for commands “CreateAlarmByName” , “CreateNoteWithNameAndContent” and “RenameEvent”
2. Modify Grammar file of one previous Command “CreateAlarmWithTime”

Just replaced <time> to <dateTime>

1. Added new concept files required for three new commands.
2. Modified “PlaceHolderDetail.txt” file to add entry for new placeHolders.

**Problem Description:**

You are working on a Natural language Processing project, where your task is to classify input utterance to one of command.

For example

|  |  |
| --- | --- |
| Command-Name | Utterance |
| CallByName | Can you please call Saurav ganguly |
| CreateAlarmWithTime | Create an alarm at 9 O’clock tomorrow morning |
| DateTimeInPlace | What is time in New Delhi |
| CreateAlarmWithName | Make an alarm with name mechanical lecture |
| RenameAlarm | Rename my breakfast alarm to good morning |
| CreateMemoWithNameAndContent | Make a shopping list note having bread butter milk |

Any sentence can be considered as combination of

1. Concept – group of phrases having same meaning
2. PlaceHolders – Place holders for a particulate type of value in a utterance

for example –

* Static data like - names of cities, contact name

(for this project we are providing text-files having finite values of these placeHolders, so these are text-file lookup for us.)

* Date – time like – 7 am, 8 pm, 3.30 am tomorrow

You need to write code for date-time matching

* OpenPhrase – place holders which can have any value for example

“Create a note named purchase items having pen pencil compass”

In above utterance – name of note i.e. “purchase items” and content of note i.e. “pen pencil compass” are openPhrase place holders, their values can be anything.

1. Key-word – All other English words

For example:

{Create\_concept} a <dateTime> calendar event named <event\_name> at <place> and invite <contact\_name>

Ex- create a tomorrow 8 pm calendar event named Samsung demo at IIt bhu and invite Ajay

Format of writing data in terms of Concept, Keyword and PlaceHolders is called **“Grammar Format”**

In above example:

{create}

* it is a concept and this concept can have values like (in “Grammar Format” we always write name of concept in curly braces)
* Create
* Make
* Set
* Get me

<dateTime>

* It is a PlaceHolder for date & time (you need to write a code to detect basic date-time format for ex. 7 am, 8 pm, 9:30 am tomorrow, today 6:30 PM, tomorrow 5 am). In Grammar we always write placeHolders in angular brackets.

<event\_name>

* It is a openPhrase type placeholder which can have any value

<contact\_name>

* It is a text-file look-up PlaceHolder and a text file having person’s name is provided in resources . You need to write code for matching text-file values with in-put utterance.

|  |
| --- |
| Saurav Ganguly |
| Sachin Tendulkar |
| Rahul Dravid |

<place>

* it is a text-file look-up PlaceHolder and a text file having place name is provided in resources . You need to write code for matching text-file values with in-put utterance.

|  |
| --- |
| Taj Mahal |
| Red fort |
| IIT BHU |

You will be given “Grammar file” for every command, “Concept file” for every concept, one text file for every text-file lookup placeholder.

you need to design and write code for pattern-matching algorithm to classifying test data in one of command, **and detect boundary of every placeholder present in input utterance.**

Test data will not be in Grammar format, it will be normal English utterance.

**You must not change content of any Resource file(Grammar file, concept file, PlaceHolders file) provided by us.**

**If your algorithm requires any extra metadata to store in file – you can add new file or folder and mention about your custom files in "readme.txt" file. All those new files or folders must be part of zip file you submit. (Please use relative paths for reading / writing files)**

**Conditions to be Handled are:-**

1. Test value for openPhrase placeHolder can be anything, matched command name and placeholder boundary must be detected correctly.

One Grammar sentence can have more than one openPhrase placeHolder

1. Some phrases or keywords can be optional at test time

* Stop words can be optional in sentence. (removal of stop words like – a, an, the etc. does not change overall meaning of sentence, so treat stop words as optional in matching algorithm)
* Pleasant phrases can be optional in test sentence

Pleasant phrases are like – “please” ,”can you please” , “thanks”, “thank you”, “would you please” etc.

1. There may be few unknown words in test data (words which are not present in Grammar, Concept, text-file based PlaceHolders), in that case your logic should return nearest match.
2. Partial-phrase matching in text-file lookup placeHolder

Let’s say In contact\_name text-file there is entry “Saurav Ganguly” and user says

Utterance – “call saurav” or “call ganguly” than also it should match to correct command.

1. Basic Date-time matching of following format (dateTime PlaceHolders should support following format)

<number> a.m.

a.m. <number>

<number> p.m.

p.m. <number>

<number> a.m. <day>

a.m. <number> <day>

<number> p.m. <day>

p.m. <number> <day>

<day> <number> a.m.

<day> a.m. <number>

<day> <number> p.m.

<day> p.m. <number>

<number> a.m. <date>

a.m. <number> <date>

<number> p.m. <date>

p.m. <number> <date>

<date> <number> a.m.

<date> a.m. <number>

<date> <number> p.m.

<date> p.m. <number>

Here

1. <number> can be any real number valid for Clock
2. <day> can be – today, tomorrow, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday
3. <date> can be a valid date in following format (date PlaceHolder should support following format)

<number>month ( 7 January , 14 February etc.)

Month <number> (March 20, October 15 etc.)

<number> Month Year ( 7 January 2018 , 14 February 2019 etc.)

Year Month <number> (2018 March 20, 2019 October 15 etc.)

For date PlaceHolder - <number> can be 1 to 31 range

**Provided Resources:**

1. In a folder “Grammar”– we are providing Grammar files. **Name of “Grammar” file should be treated as Command Name**.
2. In a folder “Concept” – we are providing concept files for every concept used in “Grammar”

Name of concept file is same a name of concept used in Grammar.

1. In a folder “PlaceHolder” – we are providing text files for contact name & places
2. In a folder “PlaceHolder” – we are providing a PlaceHolderDetail.txt – It will be having tab separated detail for placeHolders,

for ex –

|  |  |  |
| --- | --- | --- |
| Name | Type | Table / Function |
| < contact\_name > | FileLookup | contact\_name.txt |
| <place> | FileLookup | places.txt |
| <dateTime> | Codded | DateTimePlaceHolder |
| <day> | Codded | DayPlaceHolder |
| <date> | Codded | DatePlaceHolder |
| <number> | Codded | NumberPlaceHolder |
| <alarm\_name> | openPhrase | AlarmNameOpenPhrase |
| <note\_name> | openPhrase | NoteNameOpenPhrase |
| <note\_content> | openPhrase | NoteContentOpenPhrase |
| <src\_event\_name> | openPhrase | SrcEventNameOpenPhrase |
| <tgt\_event\_name> | openPhrase | TgtEventNameOpenPhrase |

Purpose of providing “PlaceHolderDetail.txt” is to avoid hard-coding as much possible.

So for processing PlaceHolder matching you can first read its Type

For text-file looup (FileLookup) you can get text file name

For Codded placeholder you can use last entry to call appropriate function in your code.

For openPhrase placeHandler you can use last entry if command specific special handling required for catching openPhrase boundary correctly.

1. Basic test case doc. Your code should read test cases from “input.txt” present in “Testing” folder

And create a “output.txt” **in “Testing” folder.**

Format of “input.txt” and “output.txt” is given below.

If command has more than one placeHolder, you can print their values in any order

1. Code must be expandable – if we add more Resources like “Grammar Files”, “Concept Files” , “text-file for placeHolder”& “PlaceHolderDetail” – new commands should get automatically enabled.
2. As NLU code should be designed to work for multiple languages (English, Spanish, Hindi etc.) , so avoid any hard-coding in the code, you can create more resource files if required.

(Adding required resource files can help you to make code – language in-dependent.

**Testing/input.txt** :

Case#1:

I want you to make tomorrow 9 O clock morning event

Case#2:

Create tomorrow 9:30 p.m. event at Bangalore

Case#3:

Can you please set a team lunch alarm at 11 am tomorrow

Case#4:

Can you make a call to Amitabh for me

Case#5:

Send an email to Rahul Dravid

Case#6:

Update me about weather in New York thanks

Case#7:

Can you tell me when will it rain in Bangalore at 6 a.m. tomorrow

Case#8:

Find out What is time in London now thanks

Case#9:

What is time now

Case#10:

What is date today

Case#11:

Can you book a flight ticket on 14 February from Bangalore to Varanasi

Case#12:

on 5 a.m. tomorrow make build body alarm please

Case#13:

with name go to make money create a alarm at 8 a.m. tomorrow

Case#14:

please make a note with title shopping list and write milk egg butter in it

Case#15:

can you please note down 100 dollar for patrol for me in trip expanse

Case#16:

would you please search fruit and juice and rename to monkey in market in event application

**Testing/Output.txt**:

Case#1:

CreateEventWithTime

<dateTime> : "tomorrow 9 O clock"

Case#2:

CreateEventWithTimeAndPlace

<dateTime> : "9:30 p.m."

<place> : "Bangalore"

Case#3:

CreateAlarmWithTime

<dateTime> : "11 am tomorrow"

Case#4:

CallByName

<contact\_name> : "Amitabh"

Case#5:

SendEmail

<contact\_name> : "Rahul Dravid"

Case#6:

WeatherInCity

<place> : "New York"

Case#7:

WeatherTimeCity

<place> : "Bangalore"

<dateTime> : "6 a.m. tomorrow"

Case#8:

DateTimeInPlace

<place> : "London"

Case#9:

DateTimeNow

Case#10:

DateTimeNow

Case#11:

BookTicket

<place> : "Bangalore"

<place> : "Varanasi"

Case#12:

CreateAlarmByName

<alarm\_name> : "build body"

Case#13:

<alarm\_name> : "go to make money"

Case#14:

CreateNoteWithNameAndContent

<note\_name> : "shopping list"

<note\_content> : "milk egg butter"

Case#15:

<note\_name> : "trip expanse"

<note\_content> : "100 dollar for patrol"

Case#16:

<src\_event\_name> : "fruit and juice"

<tgt\_event\_name> : "monkey in market"